# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**TRADE-OFF STUDY FOR THE HIT-TO-KILL INTERCEPTION OF BALLISTIC MISSILES IN THE BOOST PHASE**

by

Weng Wai Leong

December 2009

| | |
|---|---|
| Thesis Advisor: | Oleg Yakimenko |
| Second Reader | Christopher Adams |

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| | |
|---|---|
| **Report Documentation Page** | *Form Approved OMB No. 0704-0188* |

Public Reporting Burden For This Collection Of Information Is Estimated To Average 1 Hour Per Response, Including The Time For Reviewing Instruction, Searching Existing Data Sources, Gathering And Maintaining The Data Needed, And Completing And Reviewing The Collection Of Information. Send Comments Regarding This Burden Estimate Or Any Other Aspect Of This Collection Of Information, Including Suggestions For Reducing This Burden, To Washington Headquarters Services, Directorate For Information Operations And Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, And To The Office Of Management And Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

| | | |
|---|---|---|
| **1. AGENCY USE ONLY *(Leave Blank)*** | **2. REPORT DATE**<br>December 2009 | **3. REPORT TYPE AND DATES COVERED**<br>Master's Thesis |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** Trade-off Study for the Hit-to-kill Interception of Ballistic Missiles in the Boost Phase | **5. FUNDING NUMBERS** |
| **6. Author(S)** Weng Wai Leong | |
| **7. Performing Organization Name(S) And Address(Es)**<br>Naval Postgraduate School<br>Monterey, Ca 93943-5000 | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. Sponsoring /Monitoring Agency Name(S) And Address(Es)**<br>N/A | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES** The Views Expressed In This Thesis Are Those Of The Author And Do Not Reflect The Official Policy Or Position Of The Department Of Defense Or The U.S. Government.

| | |
|---|---|
| **12a. Distribution / Availability Statement**<br>Approved for public release; distribution is unlimited | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT (Maximum 200 Words)**

In recent conflicts, ballistic missiles have been used to achieve military and psychological objectives. With the proliferation of weapons of mass destruction (WMD), the threat of ballistic missiles as delivery platform for WMD is of concern. Defense against such threats becomes important.

There are different guidance laws, like pursuit and proportional navigation (PN), for the missile interception of aerial targets. A new guidance algorithm was developed by John A. Lukacs and Prof Yakimenko in 2006 to demonstrate the feasibility of intercepting a ballistic missile during the boost phase. This trajectory-shaping guidance algorithm uses the direct method of calculus of variations that maximizes the kinetic energy transfer from the interceptor to the target.

A study was conducted by applying this guidance law and examining the trade-off between the various critical parameters, like intercept geometry, time, altitude and trajectory, in the optimized solution,. It provides insights into the feasibility and limitations of this guidance. A literature review of the drag model and comparison with the compensated PN guidance was also conducted. A new induced drag model was developed for future studies. The results verified that the trajectory-shaping guidance is feasible for the interception of ballistic missiles in the boost phase for a wide range of interceptor launch locations with respect to a ballistic missile detection point.

| | |
|---|---|
| **14. Subject Terms**<br>Missile Guidance Laws, Trajectory Shaping Guidance, Trade-offs, Induced Drag Polar, Optimal Flight Path, Cost Function, Boost Phase Intercept, Intercept Geometry, Time-to-Intercept | **15. NUMBER OF PAGES**<br>135 |
| | **16. PRICE CODE** |

| | | | |
|---|---|---|---|
| **17. Security Classification Of Report**<br>Unclassified | **18. Security Classification Of This Page**<br>Unclassified | **19. Security Classification Of Abstract**<br>Unclassified | **20. LIMITATION OF ABSTRACT**<br>UU |

THIS PAGE INTENTIONALLY LEFT BLANK

# TRADE-OFF STUDY FOR THE HIT-TO-KILL INTERCEPTION OF BALLISTIC MISSILES IN THE BOOST PHASE

Weng Wai Leong
Lieutenant Colonel, Republic of Singapore Air Force
Bachelor of Engineering (Mechanical), Nanyang Techonological University, Singapore, 1993

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE**
**IN**
**MECHANICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**December 2009**

Author: Weng Wai Leong

Approved by: Prof Yakimenko
Thesis Advisor

Christopher Adams
Second Reader

Knox Milsap
Dean, Graduate School of Mechanical and Aeronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

In recent military conflicts, ballistic missiles have been used to achieve military and psychological objectives. With the proliferation of weapons of mass destruction (WMD), the growing threat of ballistic missiles being used as a delivery platform for WMD by rogue nations or militant groups becomes a concern for many countries. Defense against such threats becomes increasingly important.

There are different guidance laws for the missile interception of aerial targets. These include pursuit, proportional navigation (PN) guidance as well as its variants. A new guidance algorithm was developed by John A. Lukacs IV and Prof Yakimenko in 2006 to intercept a ballistic missile during the boost phase by a missile interceptor. This TS guidance algorithm uses the direct method of calculus of variations that maximizes the kinetic energy transfer from a surface-launched missile to a ballistic missile target.

A trade-off study was conducted by applying this guidance law in simulated ballistic missile interception. This study examines the interactions and trade-offs between the various critical parameters in the intercept solution, like the endgame intercept geometry, time-to-intercept and intercept altitude. It provides insights into the feasibility and limitations of the TS guidance algorithm. A literature review of the drag model used in the algorithm and comparison of the new guidance with the compensated PN guidance was also conducted. A new induced drag model was developed for future studies.

The results verified that the trajectory-shaping guidance is feasible for the interception of ballistic missiles in the boost phase for a wide range of interceptor launch locations with respect to a ballistic missile detection point. A better understanding of the trade-offs between the key parameters allows users to optimize the performance of this guidance.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

I. INTRODUCTION....1
A. BACKGROUND....1
B. THESIS ORGANIZATION....5

II. SIMULATION AND MODELING……7
A. DESCRIPTION OF TRAJECTORY SHAPING GUIDANCE....7
B. SIMULATION SOFTWARE ARCHITECTURE....10
C. TARGET MODELING....10
1. Ballistic Missile Physical Specifications....10
2. The Ballistic Missile Model Program....11
D. INTERCEPTOR MODELING....14
1. Basic Definitions and Assumptions....14
2. Interceptor Missile Model Program....14
E. INITIALIZATION....16
F. COMMON FUNCTIONS....16
1. SMDragC.m and BRDrag.m....16
2. STatmos.m....17
G. THE FLIGHT PROGRAM....18

III. COMPARISON OF COMPENSATED PROPORTIONAL NAVIGATION WITH TRAJECTORY-SHAPING GUIDANCE....21
A. PROPORTIONAL NAVIGATION GUIDANCE....21
B. COMPENSATED PROPORTIONAL NAVIGATION....23
C. DISADVANTAGES OF COMPENSATED PN GUIDANCE FOR BOOST PHASE INTERCEPTION....23
D. ADVANTAGES OF TRAJECTORY SHAPING GUIDANCE....24

IV. INDUCED DRAG MODEL....27
A. AERODYNAMIC DRAG....27
B. COMPONENTS OF DRAG....27
C. INDUCED DRAG POLAR....28

V. RESULTS AND DISCUSSIONS....33
A. FEASIBILITY OF THE TRAJECTORY SHAPING GUIDANCE....33
B. REGION OF POSSIBLE DEPLOYMENT POSITION OF THE INTERCEPTOR MISSILE....35
C. EFFECT OF VARYING THE INTERCEPT GEOMETRY COEFFICIENT....37
D. INTERCEPT GEOMETRY TRENDS....40

VI. CONCLUSIONS....43

APPENDIX A: TABLES AND PLOTS OF SIMULATION RESULTS....45

APPENDIX B: PARTICIPATION IN AIAA YOUNG PROFESSIONAL ROCKET LAUNCH COMPETITION IN MAY 2009....57

APPENDIX C: INDUCED DRAG PROGRAMME DEVELOPED................................69
1. ZLDragC_mod.m....................................................................69

APPENDIX D: MODELING PROGRAM CODE ..........................................71
A. 3DOF INTERCEPTOR....................................................................71
1. SMFlight3.m..........................................................................71
2. SMParam3.m..........................................................................76
3. SMDrag.m..............................................................................78
B. 3DOF TARGET ...........................................................................79
1. BRFlight3.m ..........................................................................79
2. BRParams3.m.........................................................................82
3. BRDrag.m..............................................................................83
C. GUIDANCE ALGORITHMS..........................................................84
1. SMGuidance.m.......................................................................84
2. SMGuidanceCost.m................................................................89
3. SMTrajectory.m......................................................................92
D. COMMON FUNCTION - STANDARD ATMOSPHERE.........................98
1. STatmos.m ............................................................................98

APPENDIX E: DETAILED DESCRIPTION OF TRAJECTORY SHAPING GUIDANCE (REPRODUCED FROM [10])........................................................101
A. PROBLEM STATEMENT ...........................................................101
B. CALCULUS OF VARIATIONS ....................................................102
C. PROGRAM DEVELOPMENT .......................................................106
1. Boundary Conditions..............................................................106
2. Separating and Recombining Space and Time ............................108
3. Reference Trajectory...............................................................109
4. Inverse Dynamics...................................................................111
5. Cost and Penalty Functions......................................................112

LIST OF REFERENCES .................................................................115

INITIAL DISTRIBUTION LIST ........................................................117

# LIST OF FIGURES

Figure 1. Phases of Ballistic Missile Trajectory [3] ........................................................2
Figure 2. Ballistic Missile Interception by Ground-launched Missile [5].........................3
Figure 3. TPD-2 ICBM (left) and SM-3 Standard Missile (right) [11,12].......................8
Figure 4. Reach of the TPD-2 ICBM Launched from North Korea.................................8
Figure 5. The SM-6 Standard Missile [14].......................................................................9
Figure 6. Comparison of the size of TPD-2 and M-6 Standard Missile [after 15]...........9
Figure 7. Comparison of the size of TPD-2 and M-6 Standard Missile [15] .................10
Figure 8. Ballistic Missile Flight Path [Ref ] .................................................................11
Figure 9. TPD-2 Thrust Profile (Boost Phase) [Ref ].....................................................12
Figure 10. TPD-2 Ballistic missile Mass (Boost Phase Only) .........................................13
Figure 11. TPD-2 Fly-out Range......................................................................................13
Figure 12. TPD-2 Altitude Profile for Entire Flight (left) and Boost Phase (right)..........14
Figure 13. SM-6 Interceptor Thrust Profile......................................................................15
Figure 14. SM-6 Interceptor Mass (Boost Phase Only) ...................................................16
Figure 15. Drag Coefficient by Mach Number and Flight Phase.....................................17
Figure 16. Atmospheric Temperature Variation by Altitude ............................................18
Figure 17. Atmospheric Density by Altitude ...................................................................18
Figure 18. Atmospheric Pressure Variation by Altitude ..................................................18
Figure 19. Intercept Geometry for PN Guidance [after 21] .............................................22
Figure 20. Transonic Drag Rise .......................................................................................29
Figure 21. Parabolic Drag Polar for Asymmetric Body ...................................................29
Figure 22. Parabolic Drag Polar for Symmetric Body .....................................................30
Figure 23. Parabolic Drag Polar for Different Mach Number (Symmetric Body)...........31
Figure 24. Region of Interceptor Position for Interception of Ballistic Missile...............36
Figure 25. Typical Intercept Geometry ............................................................................39
Figure 26. Illustration of Angle between interceptor Launch Position and Ballistic Missile Trajectory Plane ....................................................................41
Figure 27. Effect of Interceptor Launch Direction on Intercept Geometry......................41
Figure 28. Team Members and Advisor with Their Rocket at the Launch Site...............57
Figure 29. Timeline for the Rocket Launch Project ........................................................60
Figure 30. The Specifications and Dimensions of the Quasar Rocket Kit.......................61
Figure 31. The Specifications and Dimensions of the Quasar Rocket Kit.......................62
Figure 32. The Specifications and Dimensions of the Quasar Rocket Kit.......................63
Figure 33. Team Members and Advisor with Their Rocket at the Launch Site...............64
Figure 34. Team Members and Advisor with Their Rocket at the Launch Site...............65
Figure 35. Team Members and Advisor with Their Rocket at the Launch Site...............66
Figure 36. Official Altitude Plot for Team Peacock ........................................................67
Figure 37. Variation of Path with $\tau_f$ and $x_{10}'''$ (after [Ref 18])........................................105
Figure 38. Variation of First Derivative of Path with $\tau_f$ and $x_{10}'''$ ...................................106

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

Table 1. Specifications of TPD-2 Ballistic Missile.......................................................11

Table 2. Specifications of SM-6 Interceptor Missile.....................................................14

Table 3. WGS-84 Values...........................................................................................16

Table 4. No of Iterations Required for Feasible Interceptor Solution (WIA = 0)..........34

Table 5. No of Iterations Required for Feasible Interceptor Solution (WIA = 100)......34

Table 6. No of Iterations Required for Feasible Interceptor Solution ($W_{IA}$ = 1000).....35

Table 7. Time-To-Intercept, Altitude And Impact Angle Deviation For Different Intercept Geometry Coefficient, $W_{IA}$................................................................38

Table 8. Budget for Rocket Launch Competition..........................................................59

Table 9. Interceptor Known Data...................................................................................107

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| 3-D | 3-Dimentional |
| 3DoF | 3 Degree-of-Freedom |
| AIAA | American Institute of Aeronautical and Astronautics |
| BMDS | Ballistic Missile Defense System |
| CG | Centre of Gravity |
| CP | Centre of Pressure |
| DoD | Department of Defense |
| DPRK | People’s Democratic Republic of Korea |
| EFC | Electronic Forward Closure |
| ERAM | Extended Range Anti-Air Warfare missile |
| ICBM | Inter-Continental Ballistic Missile |
| LOS | Line-of-Sight |
| PN | Proportional Navigation |
| TPD-2 | Taepo-Dong 2 Ballistic Missile |
| SM-6 | Standard Missile - 6 |
| TS | Trajectory Shaping |
| WMD | Weapons of Mass Destruction |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

The author wishes to acknowledge the patience and understanding of his wife, Lee Ling, and his two lovely girls, Kelly and Jolee, for my not being able to spend enough time with them while working on this thesis, both at home and in school.

The author would also like to thank his thesis advisor for his guidance and advice in the project. His helpfulness, understanding and friendship have made the whole journey such a positive experience for the author.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. BACKGROUND

In recent conflicts, notably the first Gulf War in 1992 and the Israeli-Lebanon conflict in 2007, the use of tactical ballistic missiles to achieve military and psychological objectives had alerted political and military leaders around the world of the immense threat posed by such weapons. With the proliferation of weapons of mass destruction (WMD), there has been a greater concern that tactical ballistic missiles will be used by rogue nations or militant groups, as a delivery platform for such weapons [1]. The outcome can be devastating, with the loss of many innocent lives and large-scale destruction. Many countries have invested substantially in developing ballistic missile defense systems as a strategy against such threats. In particular, the United States Department of Defense (DoD) was directed to develop a conceptual framework in 2002—the Ballistic Missile Defense System (BMDS), in which active defense of intercepting incoming ballistic missiles in all phases of its trajectory, is a major strategy [2]. An integrated ballistic missile defense system was subsequently developed based on a layered, defense-in-depth strategy. In order to intercept a ballistic missile, flying at supersonic speed and high altitude, the interceptor must have the required response time, speed and accuracy. The interception can take place in any of the three distinct phases—boost, midcourse or terminal, as illustrated in Figure 1.

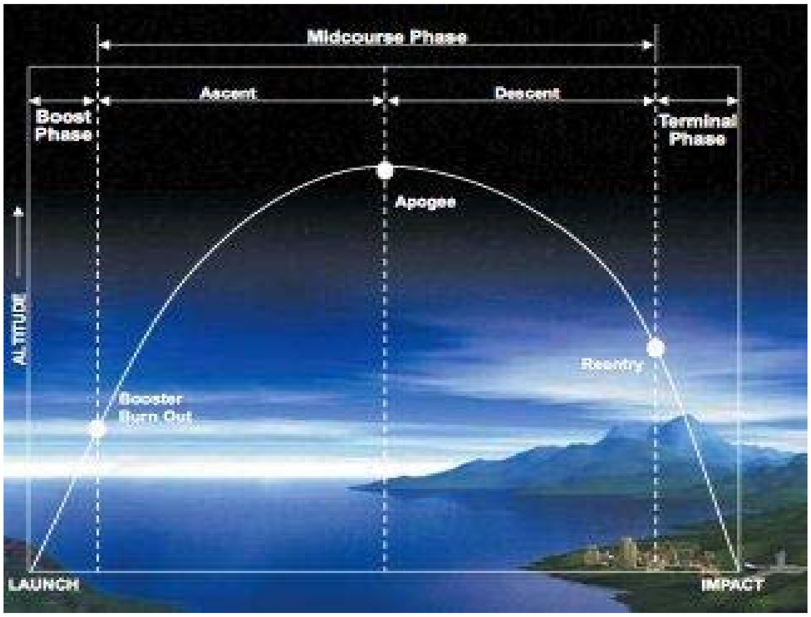


Figure 1. Phases of Ballistic Missile Trajectory [3]

During the boost phase, the ballistic missile rocket motors are firing in order to accelerate the missile to a high trajectory altitude. The advantages of boost phase interception are that the hot and bright rocket exhaust plume makes detection and targeting easier, and decoys cannot be used during this phase. The disadvantages lie in the geographical siting of the interceptor system (which has to be close to hostile territory) and the short time to intercept, typically about 180 seconds. Figure 2 shows images of ballistic missile interception by ground-launched intercept missile. In some literatures, defense analysts and scientists believed that interception of ballistic missile by surface-launched interceptor using established guidance laws is not feasible. However, the trajectory-shaping guidance developed by Lukacs and Prof Yakimenko in 2006 [4] shows that boost phase interception is possible, with promising results on the effectiveness of such interception, subject to limitations.

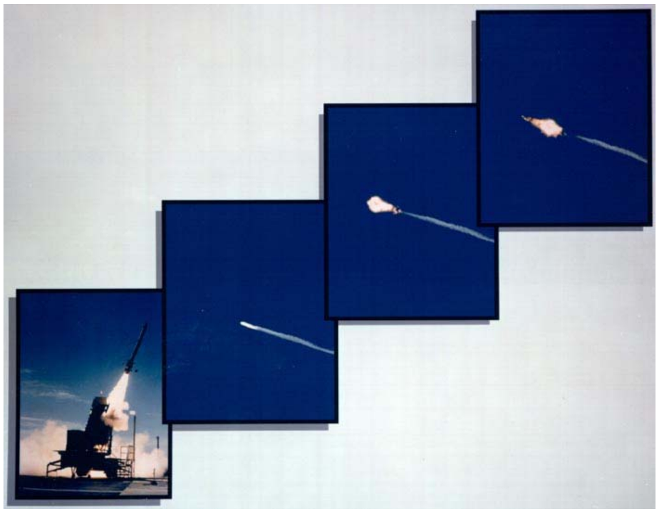


Figure 2. Ballistic Missile Interception by Ground-launched Missile [5]

In the mid-course phase, the ballistic missile is in space after the rocket burns out. The coast period through space before re-entering the atmosphere can take several minutes, up to 20 minutes for a long-range Inter-Continental Ballistic Missile (ICBM). The advantage of intercepting a ballistic missile in this phase is that there will be sufficient decision and intercept time as well as a greater flexibility in the geographical defensive position of the interceptor system. However, such interception would require a larger and, hence, heavier interceptor missile, complemented by sophisticated radar and other sensors to handle potential space-based decoys.

In order to intercept a ballistic missile in the terminal phase, the interceptor missile would have to do so after the ballistic missile re-enters the atmosphere. The advantages lie in the requirement for smaller and lighter interceptor missile, less sophisticated radar and lower possibility of decoy as they are not likely to work in this phase. The disadvantages are the very short reaction time (in the region of 30 seconds or less), less defended geographical coverage and possible effect of hazardous materials over the target area in the case of detonation of chemical, biological or nuclear warhead(s) mid-air.

This paper focuses on the interception of ballistic missiles in the boost phase. Intercepting a ballistic missile in its boost phase is the ideal solution for missile defense, since the missile is most vulnerable during this phase of its flight. This is done usually over the launch territory. There are, however, many challenges associated with this phase. The interceptor will have to contend with large acceleration rates, very short reaction time and requires reliable scanning and tracking [5,6]. There are several systems are under development for conducting boost phase interception, they include ground-based missiles, airborne lasers and space-based intercept missiles.

This study only looks at surface-based missile interceptor. The missile guidance law is one of the most important factors that determines the feasibility and effectiveness of the intercept solution. If intercept is possible, then the next step is the interceptor's capability to kill the target. With a very short engagement time, the interceptor missile needs a high speed and energy to reach the target. This imposes a limitation on the size of warhead. Earlier concepts and studies recognized that that a simple warhead effect is not sufficient to destroy an ICBM, hence the initiation of the development of hit-to-kill technologies [7,8]. This concept relies on relatively smaller high speed missile, guided accurately to the target and transferring maximum kinetic energy to the ballistic missile for a kill. It suggests some form of geometry control during impact.

The actual impact geometry is not an important parameter for most guidance laws implemented in intercept missiles which uses warhead effect (proximity fuze) as their main kill mechanism. However, for a hit-to-kill endgame condition which demands for maximum transfer of kinetic energy to the target missile, the intercept geometry commands a greater attention and has to be controlled as an input to the guidance law.

The objective of this thesis is to conduct a trade-off study to evaluate the effectiveness of the hit-to-kill trajectory-shaping guidance through simulation, using the guidance algorithm developed by Lukacs and Prof Yakimenko in 2006. The code developed for the guidance law will "generate the interceptor's entire flight path in order to minimize the distance traveled, minimize the time to intercept, and maximize kinetic energy transfer by controlling the interception geometry while providing near-optimal flight path to interception.  This will be done by utilizing the direct method of calculus of variations combined with inverse dynamics theory to reverse engineer in real time, an optimal flight path using the missile's onboard sensors and computers". [9]

## B. THESIS ORGANIZATION

This thesis consists of six chapters. Chapter I provides a brief introduction and overview of ballistic missiles interception in the boost phase and outlines the objectives of this thesis.

Chapter II provides a literature review of the models and guidance program, developed by Lukacs and Prof Yakimenko in 2006, based on the trajectory-shaping guidance law. The trade-off study is based on this guidance algorithm, with only slight modification in certain areas, for the simulation. For completeness, it is instructive to provide a brief overview of the guidance law, key characteristics and the MATLAB program developed as a prelude to the trade-off study and discussion. The detailed description of the theory behind the guidance algorithm and models used is appended in Appendix E. Much of the content in this chapter is extracted from the thesis written by Lukacs [10].

Chapter III discusses an alternative guidance law—the well-established Proportional Navigation (PN) guidance that is used in many advance missiles currently in service. It provides some background information on PN guidance and makes some comparison with the trajectory-shaping guidance.

Chapter IV describes the drag model that is used in the trajectory-shaping guidance code and presents the need to incorporate an induced drag model as an enhancement to the existing model for better estimation of drag on both the ballistic and interceptor missile.

Chapter V summarizes the results and discusses the feasibility of employing such guidance in a real-world scenario.

The Appendices include data and plots of the simulation results obtained, report on the author's participation in the AIAA Rocket Launch Competition, MATLAB code for the induced drag model, detailed description of the trajectory-shaping guidance and the complete MATLAB program codes.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. SIMULATION AND MODELING

## A. DESCRIPTION OF TRAJECTORY SHAPING GUIDANCE

In 2006, Lukacs and Prof Yakimenko [9] developed a simulation code in MATLAB as a demonstration of the feasibility of intercepting a ballistic missile during the boost phase by a surface-launched interceptor missile using the Trajectory Shaping guidance law (henceforth termed as TS guidance in the report).

The TS guidance uses the principle of flight path optimization from the interceptor to the predicted target position. It relies on high-order polynomial as a reference function for the flight path and uses virtual as opposed to physical domain in optimization process. A preset thrust history is used in the computation of interceptor flight path. The flight path is derived by minimization a combined performance index including intercept geometry, time-to-intercept, penalty on altitude and dynamic constraints. The performance index, J is expressed in the following equation,

$$J = t_{go}^{2} + W_{IA}(\mu - \mu_{desired})^{2} + P_1 + P_2$$

where $t_{go}$ is the time-to-intercept, $\mu$ is the impact angle, $P_1$ is the penalty on intercept altitude and $P_2$ is the penalty on dynamic constraints. The application of such guidance is particularly suitable for interception of targets with relatively fixed predicted trajectory, like that of a ballistic missile in the boost phase.

In the simulation, the ballistic missile target was modeled after the Taepo-Dong 2 (TPD-2) ballistic missile developed by the People’s Democratic Republic of Korea (PDRK, also known as North Korea) while the U.S.-made SM-6 Standard missile, an upgraded and extended range version of the SM-3 (shown in Figure 3) was used as the interceptor missile. A 3 degree-of-freedom (3DoF) mathematical model was previously developed and used to simulate the trajectory and flight characteristics of both the ballistic and interceptor missile based on available missile data from the open source. The intercept path is continuously calculated onboard the interceptor missile as a two-point boundary value problem, using Direct Methods of Calculus of Variation to calculate a near-optimal flight path and the control commands necessary to achieve it (refer to Appendix 3 for the detailed description of the TS guidance).

Figure 3. TPD-2 ICBM (left) and SM-3 Standard Missile (right) [11,12]

The TPD-2 ballistic missile, developed by North Korea is believed to be an intercontinental ballistic missile (ICBM) capable of an effective range of 6000 to 6500 km, with a 3-stage booster [13]. This put Alaska and all Asia Pacific countries up to northern Australia within reach of the TPD-2, from launch sites within North Korea [see Figure 4]. In order to have a feasible solution of intercepting this missile during its boost phase, the interceptor must be fired from a ship or land-based launcher within range of the ballistic missile launch site.

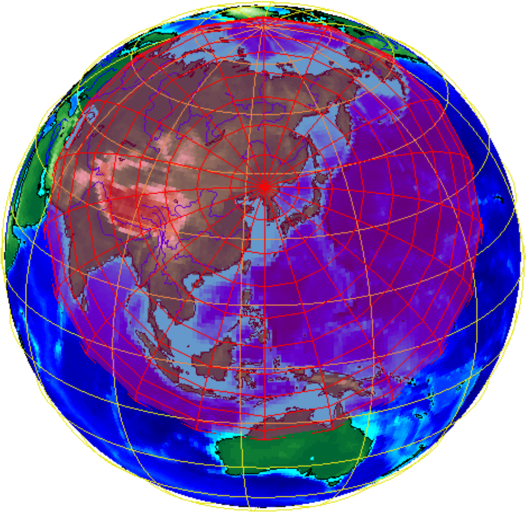


Figure 4. Reach of the TPD-2 ICBM Launched from North Korea

The SM-6 (shown in Figure 5) is the U.S. Navy's latest Extended Range Anti-Air Warfare missile (ERAM) that employs active-homing terminal guidance—a key feature that allows for accuracy necessary to intercept an ICBM in the boost phase. The SM-6 missile is the upgraded and extended range version of the SM-3, currently under development by Raytheon, for the U.S. Navy as the next generation anti-ballistic missile defense system.  It is designed to be capable of intercepting ballistic missiles in the boost phase if deployed within range of the ballistic missile launch site and complemented by suitable sensor systems to detect the launch. Figure 6 shows the comparison of the size of the TPD-2 and SM-6.

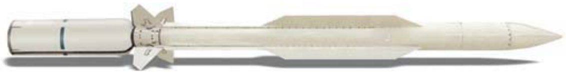


Figure 5. The SM-6 Standard Missile [14]

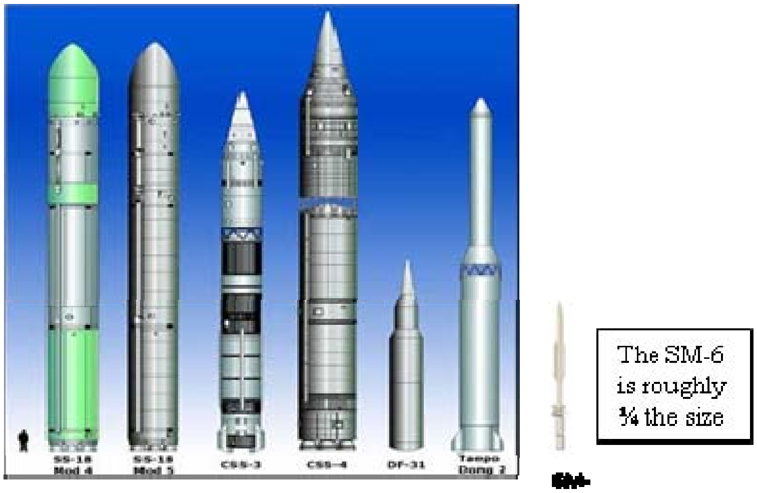


Figure 6. Comparison of the size of TPD-2 and M-6 Standard Missile [after 15]

This chapter provides a brief overview of the 3-dimensional (3D) target model that operates in the Earth’s gravitational field, using available TPD-2 data from the open source. A 3-D interceptor model, based on the SM-6 specification, was developed that operates in the Earth’s gravitational field, was also used. Brief description of the various function files, assumptions made, data and values used in the simulation study is also presented to help the reader understand the program flow. A listing of the MATLAB program code as well as detailed description and theory of the various functions are reproduced in Appendices D and E, respectively, for completeness and easy cross-reference.

## B. SIMULATION SOFTWARE ARCHITECTURE

The general program flow of the 3DoF simulation conducted is shown in Figure 7. The remaining sections of this chapter will briefly describe the function of the main blocks shown in the architecture, models used and the general flow of the simulation.

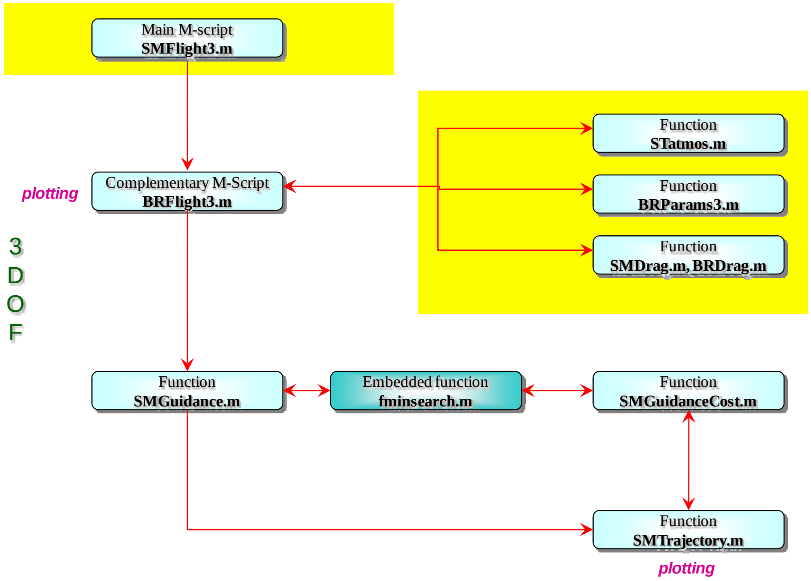


Figure 7. Comparison of the size of TPD-2 and M-6 Standard Missile [15]

## C. TARGET MODELING

### 1. Ballistic Missile Physical Specifications

Table 1 shows the physical characteristics and specification of the TPD-2 missile [13].

| | Overall |
|---|---|
| Length | 32 m |
| Payload | 750-1000 kg |
| Range | 3500–4300 km |
| Stages | 2 |

| | Stage 1 | Stage 2 |
|---|---|---|
| Diameter | 2.2 m | 1.335 m |
| Length | 16 m | 14 m |
| Launch Weight | ~60,000 kg | 15,200 kg |
| Thrust | ~103,000 kg$_f$ | 13,350 kg$_f$ |

| | |
|---|---|
| Thrust Chambers | 4,1 |
| Type | LR ICBM |

| | | |
|---|---|---|
| Fuel / Oxidizer | TM-185 / AK-27I | TM-185 / AK-27I |
| Propellant Mass | - | 12,912 kg |
| Burn Time | ~125 s | 110 s |

Table 1. Specifications of TPD-2 Ballistic Missile

### 2. The Ballistic Missile Model Program

The 3DoF model developed by Lukacs and Prof Yakimenko in 2006 comprises a series of MATLAB functions on an iterative integration loop, using 4 function files to accomplish the modeling. A brief description of each function is as follows:

1. BRFlight3.m - integrates each time step to determine the current position, attitude, and aerodynamic forces acting on the rocket/missile;
2. BRParams3.m - determines the mass of the ballistic missile and the surface reference area;
3. BRDrag.m - determines the drag coefficient;
4. STatmos.m – determines the properties of the local atmosphere;

The program BRFlight3.m generates a ballistic flight path of the ballistic missile target based on the model developed by Zarchan [16]. The program generates a 3-D flight path that is contained within the 2-D x-z plane shown in Figure 7, where the asterisks represent the staging events.

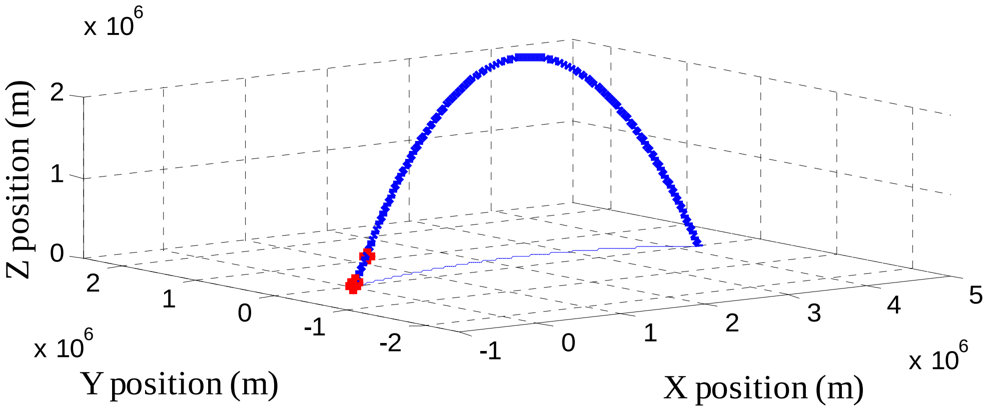


Figure 8. Ballistic Missile Flight Path [Ref ]

The launch position and angle are as follows:

$$x_0 = 0$$
$$y_0 = 0$$
$$z_0 = \mathrm{Re}$$
$$\Theta = 85^{o}$$

where Re is the radius of the Earth (6,378,137 m) based on WGS-84 system.

The thrust is assumed fixed for each stage of the propulsion. At 130 and 240 seconds after launch, the thrust drops instantaneous to represent the staging events. At completion of the boost phase, the thrust is zero. The thrust profile of the two stages is shown in Figure 8.

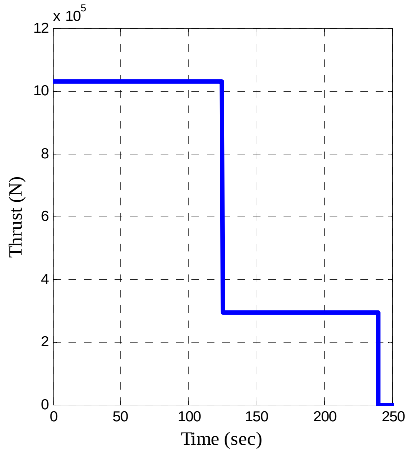


Figure 9. TPD-2 Thrust Profile (Boost Phase) [Ref ]

In order to account for the aerodynamic forces, drag has to be calculated. Atmospheric conditions, like density and temperature, are first determined using the function STatmos.m. The missile drag coefficient, CD is then computed by the function BRDragC.m. This is followed by calculating the drag force.

The ballistic missile's mass is a simple function of time (as shown in Figure 9) and this is computed using the function BRParams3.m. The mass drops sharply during the staging events at 130 and 240 seconds. After the completion of the boost phase, the mass remains constant for the remaining duration of the flight.

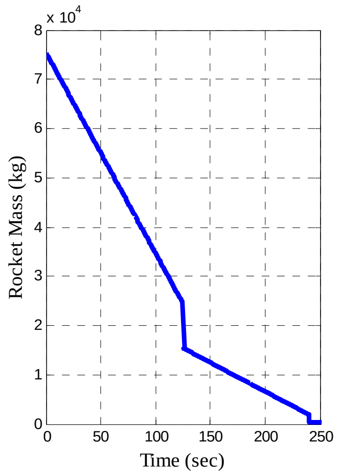


Figure 10. TPD-2 Ballistic missile Mass (Boost Phase Only)

The final step of the ballistic missile model is to record all the data for the ballistic missile. This data will be called by the interceptor model to simulate detection by the missile's onboard sensors. The interceptor missile will then register the location and velocity of the target missile at the appropriate intervals. In this program, a 60-second delay is assumed for the launch of the interceptor missile to account for the detection and decision loop.

The final ballistic fly-out range and the altitude profile predicted by the model is presented in Figure 10 and 11.

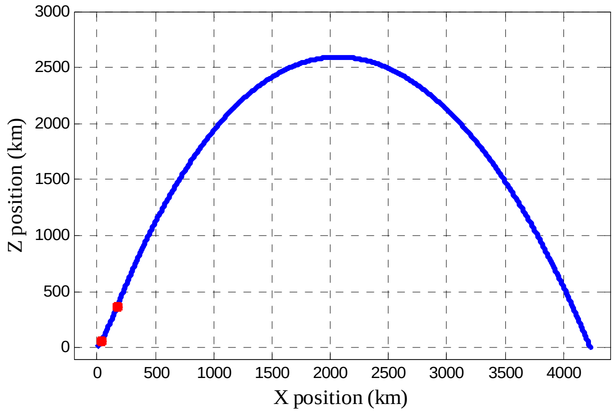


Figure 11. TPD-2 Fly-out Range

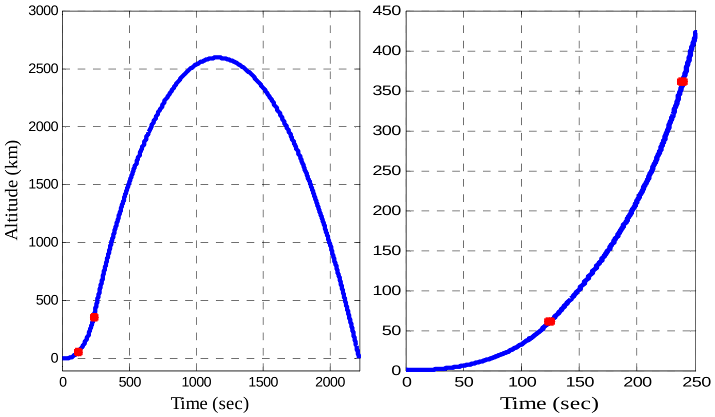


Figure 12. TPD-2 Altitude Profile for Entire Flight (left) and Boost Phase (right)

## D. INTERCEPTOR MODELING

### 1. Basic Definitions and Assumptions

The basic specifications of the SM-6 missile are presented in Table 2 [17].

| | Overall |
|---|---|
| Length | 6.5 m |
| Payload | 115 kg |
| Range | 150 km |
| Stages | 2 |
| Thrust Chambers | 1,1 |
| Type | ERAAW |

| | Stage 1 | Stage 2 |
|---|---|---|
| Diameter | 0.53 m | 0.34 m |
| Length | 1.72 m | 4.78 m |
| Launch Weight | 712 kg | 686 kg |
| Thrust | - | - |
| Fuel / Oxidizer | HTPB-AP | TP-H1205/6 |
| Propellant Mass | 468 kg | 360 kg |
| Burn Time | 6 s | - |

Table 2. Specifications of SM-6 Interceptor Missile

### 2. Interceptor Missile Model Program

The 3DOF model used in the program comprises 4 MATLAB functions files ran on a repeating integration loop. A brief description of the function files are as follows:

1. SMFlight3.m - integrates each time step to determine the current position, attitude, and aerodynamic forces acting on the missile;

2. SMParams3.m - determines the mass of the interceptor missile and the surface reference area;

3. SMDrag.m - determines the drag coefficient;

4. STatmos.m – determines the properties of the local atmosphere.

Drag is calculated in a similar fashion as that implemented in the ballistic missile model. The SMParams.m function uses the same methodology as the BRParams.m function to calculate the reference surface area and mass.

The thrust profile for the interceptor missile is also a two-step curve, just like the ballistic missile profile. The only difference is the timing for the staging event. This is shown in Figure 11. The staging event occurs at 6 and 26 seconds.

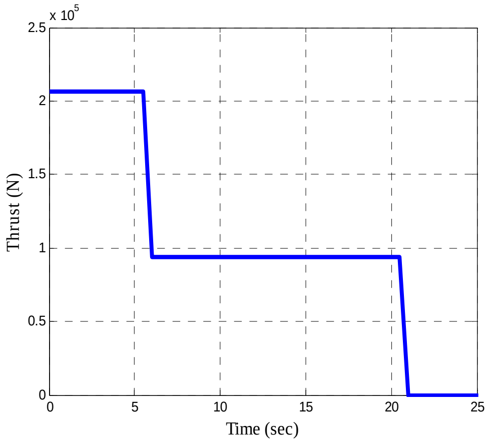


Figure 13. SM-6 Interceptor Thrust Profile

The mass of the interceptor mission changes at 6 and 26 seconds to represent the staging events. Figure 13 shows the mass profile. It shows a distinct drop at 6s and a constant mass after 26s.

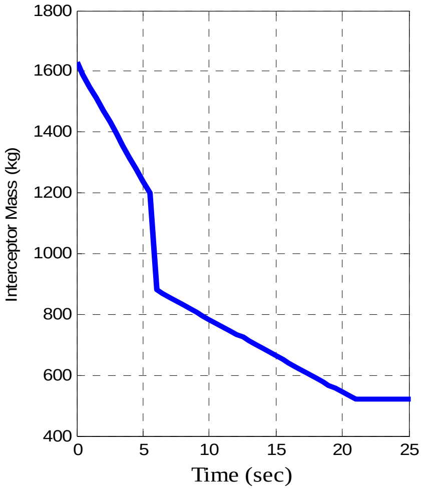


Figure 14. SM-6 Interceptor Mass (Boost Phase Only)

## E. INITIALIZATION

For initialization, the earth geographical data required was based on the WGS84 values presented in Table 3. [18]:

| | |
|---|---|
| Earth's Radius, $R_e$ | 6,378,137 m |
| Earth's Semi-Minor Axis, *b* | 6,356,752 m |
| Earth's Flattening (1/Ellipticity), *f* | 1/298.257223563 |
| Earth's Rotation Rate ($\Omega z_E$) | 7.292116 e-5 rad/sec |
| Earth's Gravitational Constant, GM | 3.986004418e14 $m^3/s^2$ |

Table 3. WGS-84 Values

## F. COMMON FUNCTIONS

Two functions were commonly by all the models. They are:

### 1. SMDragC.m and BRDrag.m

The drag on the missile is dependent on two conditions, the Mach number and the phase the missile—boost or glide. Mach number represents the speed of missile and the phase will determine the presence or absence of the base drag. The plot of drag coefficient with Mach number of the missiles is shown in Figure 14 [19]. The same function is applied through two drag files in the interceptor and ballistic missile model respectively.

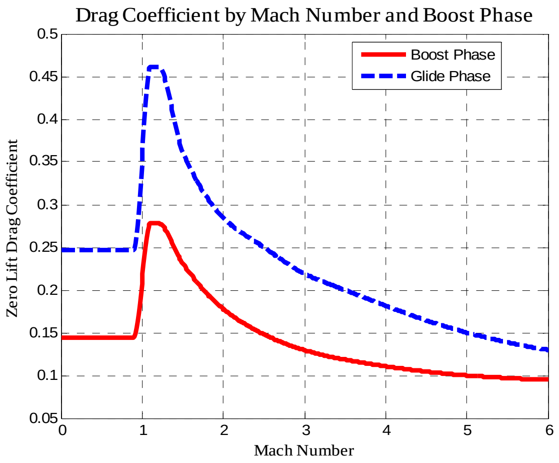


Figure 15. Drag Coefficient by Mach Number and Flight Phase

Having obtained the drag coefficient and computed the dynamic pressure computed, drag can be calculated.

### 2. STatmos.m

The atmospheric parameters, which are dependent on altitude, are computed by STatmos.m function file. The calculation was based on the 1976 standard atmospheric survey, and includes values up to 86 km in a tabular format. Figures 15–17 show the atmospheric charts used by STatmos.m to derive all the parameters required.

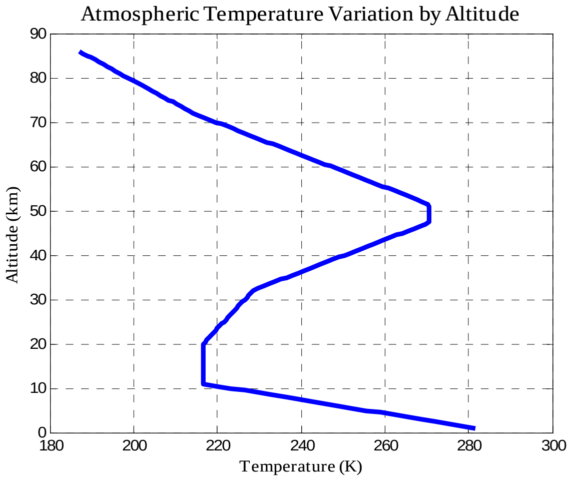


Figure 16. Atmospheric Temperature Variation by Altitude

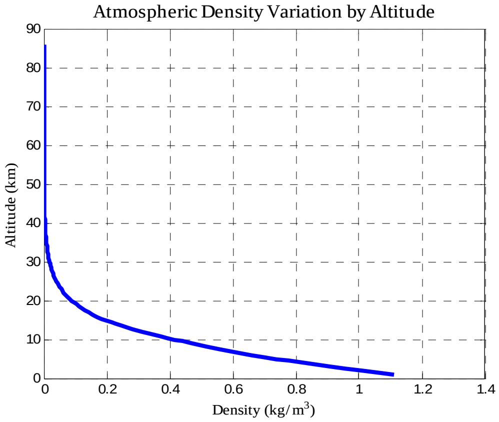


Figure 17. Atmospheric Density by Altitude

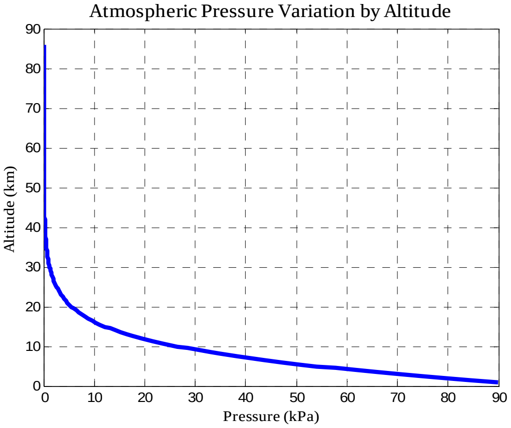


Figure 18. Atmospheric Pressure Variation by Altitude

## G. THE FLIGHT PROGRAM

Once the previous iteration (or the initialization) has been integrated, it is returned to the program as the current values. The program then uses these values to calculate all the descriptive values of the system, apply the corrective time- and position- dependant factors, and calculate the derivatives for the next iteration.

The BRParams.m and SMParams.m functions were called to determine the reference areas for the control surfaces and missile plan form areas. The program then calls the function ACoeff.m to determine several necessary aerodynamic coefficients. The function inputs are angle of attack, altitude, and pitch control surface deflection.

The respective missile model will call the function SMDrag.m and BRDrag.m to determine the value of the drag coefficient. Drag and thrust are then calculated and the execution is returned to the main program.

The trajectory of the interceptor missile is generated by the function SMTrajectory.m by applying the TS guidance law through the SMGuidance.m function. This is an iterative process, starting with a 'guess' of the interceptor final states and subsequently performing trajectory optimization by calling sub-routine SMGuidanceCost.m to minimize the cost and penalty of each iterative flight path generated. The optimized flight path is then returned to SMTrajectory for execution

THIS PAGE INTENTIONALLY LEFT BLANK

# III. COMPARISON OF COMPENSATED PROPORTIONAL NAVIGATION WITH TRAJECTORY-SHAPING GUIDANCE

In order to understand the guidance laws, it is useful to understand what "guidance" is. Guidance is the logic that issues steering commands to the vehicle to accomplish certain flight objectives' [20]. In the case of a missile and its target, it is the algorithm the missile uses to guide itself in an intercept path with the target, through autopilot commands given to its control surfaces, like fins or wings. In current interceptor missiles being fielded or deployed, a number of guidance laws were used. The type of guidance law implemented depends on the mission and the type of targets the missile is designed for. Examples of better-known guidance laws are the Beam Rider, Pure Pursuit, Proportional Navigation and its variants.

## A. PROPORTIONAL NAVIGATION GUIDANCE

PN guidance or its variants, like augmented or compensated PN, are the most common guidance law implemented in modern missiles. PN guidance has proven itself in many operations to be effective against maneuvering target, especially in an air-to-air duel or surface-to-air interception. It can be found in many air-to-air, air-to-surface and surface-to-air applications. In proportional navigation, the fundamental principle lies in the rate of change of the missile heading being kept proportional to the rate of rotation of the line-of-sight (LOS) from the missile to the target. The guidance system points the differential velocity vector at the target [7]. Figure 18 illustrates the geometry of interception using the PN guidance.

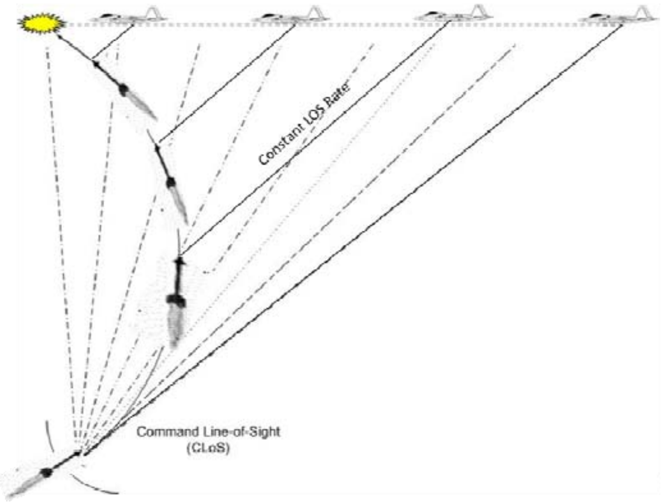


Figure 19. Intercept Geometry for PN Guidance [after 21]

In order for interception to occur, the missile heading angle, Ψ and the LOS angle, λ, must be kept constant. If the target increases speed, then λ will increase and the interceptor must correspondingly increase Ψ to maintain its interception course. The rate of change of both angles must be proportional, given by the expression [16],

$$\dot{\Psi} = N\dot{\lambda} \tag{III.A.1}$$

where $N$ is the proportionality constant, which depends on the target lead factor of the interceptor (usually between the values of 3-5 depending on the interceptor missile's maneuverability). The interceptor will maneuver until $\dot{\lambda} = 0$ (no more changes in LOS rate with the target), at which point $\dot{\Psi} = 0$. The interceptor continuously adjusts itself to maintain the above condition till interception (or a miss) occurs.

During the intercept process, the missile seeker continuously points itself at the target to derive the LOS rate and provides a signal to the guidance computer. The guidance computer in turn will convert the feedback into an acceleration command to the control surfaces to physically steer the missile. The normal acceleration to the missile's velocity vector is given by

$$a = \vec{V}^D \dot{\Psi} \tag{III.A.2}$$

which can be correlated to the LOS rate

$$a = N\vec{V}^D \dot{\lambda} \tag{III.A.3}$$

This is applicable to a 2-D case. Zipfel [20] develops a 3-D dimensional PN guidance law with the expression

$$a = N\vec{V}^{D}\Omega^{OE}u_{v} - g \tag{III.A.4}$$

where $\Omega^{VE}$ is the cross product of the LOS frame with respect to the inertia Earth frame, $u_v$ is the unit vector of $\vec{V}^{D}$, and $g$ is the added gravity bias, which counteracts the sagging tendency of the trajectory under seeker control. This form of guidance is termed by Zipfel as Pure PN.

## B. COMPENSATED PROPORTIONAL NAVIGATION

Pure PN, however, is not commonly implemented in its original form. The thrust generated by the interceptor’s propulsion creates a parasitic acceleration in the LOS angle that has to be compensated in the autopilot command. Guidance in this form is termed compensated PN. The parasitic acceleration projected onto the LOS plane can be expressed as follows:

$$\left[a_{m}\right]^{LOS} = {}^{LOS}_{\;\;w}R\left[a_{m}\right]^{w} \tag{III.B.1}$$

It is then subtracted from the PN command in equation (3.B.4) to obtain the compensated command [20]

$$a = N\vec{V}^{D}\left[\Omega^{OE}\right]^{w}\left[u_{v}\right]^{w} - {}^{LOS}_{\;\;w}R\left[a_{m}\right]^{w} - {}^{w}_{n}R\left[g\right]^{n} \tag{III.B.2}$$

where the rotation matrix ${}^{LOS}_{\;\;w}R$ of the LOS coordinates with respect to the wind frame is defined by the azimuth and elevation angles from the LOS vector.

## C. DISADVANTAGES OF COMPENSATED PN GUIDANCE FOR BOOST PHASE INTERCEPTION

Three disadvantages of the compensated PN for the boost phase interception of ballistic missiles were identified by Lukacs and Prof Yakimenko. The first disadvantage is the inherent control system time constant of the PN guidance, which utilizes current target information in a homing guidance loop with feedback control. Such control loops will inevitably result in a finite time constant which can result in considerable miss distance. The second is the disregard of the end-game environment. The PN guidance law is focused on maintaining the LOS rate constant continuously based on current missile and target parameters. However, the guidance law does not attempt to optimize missile speed and altitude, which is critical for the endgame condition, to ensure that the missile interceptor has sufficient acceleration left to intercept the target. Rather, PN guidance adopts the most direct and minimum acceleration collision path with the hope that there is little target maneuver and hence sufficient acceleration left in the interceptor for the endgame. Lastly, the intercept geometry is not controlled for the case of PN guidance, which is left to the relative speeds of the interceptor and target as well as the endgame maneuvers involved Controlling the intercept geometry allows for the maximization of kinetic energy transfer to the target, which is critical for boost phase intercept of ballistic missile since a lightweight (very small or no warhead), high speed missile with sufficient range (most of the weight goes to the fuel) is required.

## D. ADVANTAGES OF TRAJECTORY SHAPING GUIDANCE

In TS guidance, the derivation of the intercept solution - the optimized interceptor flight path, results in a set of functions, which occurs through a Cost Function and a Penalty function, which must be minimized through multiple iterations. The three parameters represented in the Cost Function are the length of the virtual arc (proportional to the flight path distance), the time to intercept and the impact angle of the final intercept (angular deviation from desired $90^o$ impact). Each of the parameters needs to be carefully weighted to correctly reflect the desired condition of the intercept, and affects the Cost Function value as a whole.

The penalty function consists of the maximum acceleration in the y- and z-directions. They represent the ‘penalty’ to pay when certain physical limitations are reached or violated. The penalty function has been set to the certain values, which are dependent on speed and altitude of the interceptor, to represent the physical limits beyond which the intercept solution will not be feasible.

The flight path optimization is done by solving a non-linear programming problem numerically real-time and once the minimum function is obtained, the algorithm will return the required control time history to the missile guidance system, which can then execute the commands and fly the derived flight path. Since the missile system can be programmed with sufficient data to compensate for its control system time constant, the system lag can be effectively negated, thus eliminating a source of error. The guidance system can be updated every few seconds to increase the accuracy of the intercept.

The main advantages of this guidance are three-fold—(1) the relatively short computation time to iterate and converge to an optimized solution, (2) the cost and penalty functions are scalable as desired to fit the mission profile and (3) elimination of the control system time constant. For a boost phase intercept mission, the TS guidance is able to address the disadvantages of the computed PN guidance and offers greater flexibility in being able to 'customize' the guidance to improve its performance to meet the different operational demands.

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. INDUCED DRAG MODEL

## A. AERODYNAMIC DRAG

The modeling of aerodynamic forces acting on a missile is a necessary step in the simulation of the missile interception of an aerial target. Drag, which represents the total sum of forces that opposes the thrust provided by the missile propulsion system, is a ‘major design parameter in satisfying the flight range requirement of tactical missiles’ [22]. This is especially so for supersonic missile. It is part of the guidance algorithm to incorporate a drag model to account for the effect of drag on the missile, which affect flight performance in all phases. A suitable drag model will enhance the accuracy of the simulation result and help to provide a more realistic prediction of the flight characteristic and performance.

## B. COMPONENTS OF DRAG

In the earth atmosphere, any flying object will experience drag. It is the force that opposes thrust (or the forward motion) of aircraft, missiles or other flying platforms. The overall drag on a body is made up of several components—friction, base, wave (or pressure) and induced drag. Friction drag refers to the drag generated by the skin friction between tangential air flow and the moving body. It is primarily a function of the body fineness ratio or length-to-diameter ratio of the missile body. Base drag accounts for the pressure difference of air flow between the nose and the base of the body. It is particularly significant during coast or glide phase of a missile since the propulsion system is no longer generating thrust. Wave drag is caused by the effect of normal pressure of air and largely dependent on the shape of the nose. The nose fineness ratio gives a measure of the nose shape, which affects the wave drag. A sharp nose will create less drag than a blunt one. Last but not least is induced drag. This is the component of drag that is ‘induced’ by the lift generated for level flight or climb. It is directly proportional to the lift. Hence induced drag is present in all flying platform as long as lift is generated by the control surfaces and in some condition, is a key contributor to the overall drag.

In mathematical terms, drag is a function of the drag coefficient, dynamic pressure and reference area, given by this equation

$$D = C_D q S_{ref} \qquad \text{(IV.B.1)}$$

The drag coefficient, $C_D$ is usually derived from empirical equation as a function of the speed of sound or Mach Number (M) in the operating regime, the angle of attack, α, which is the angle of inclination between the missile body axis and its velocity vector. In the case of a missile, the reference area is commonly taken as the cross-sectional area.

The atmospheric dynamic pressure, q is given by

$$q = \frac{\rho V^2}{2} \qquad \text{(IV.B.1)}$$

where ρ is the density of air and V is the speed of the missile.

## C. INDUCED DRAG POLAR

As briefly described in Chapter 2, for the new TS guidance, a standard drag model was used whereby the equation (IV.B.1) was used to calculate the drag. Noticed it was mentioned that $C_D$ is usually obtained empirically and different missile design is likely to have different drag characteristics. The accuracy of the drag prediction can be enhanced with the use of an induced drag model. This model predicts drag by involving the lift coefficient, $C_L$ which describes the lift generated by the missile. The lift force, L is normal to the missile velocity vector and hence also normal to drag, D (which is opposite to the velocity vector). The lift and drag coefficient can be expressed as

$$C_L = \frac{L}{qS_{ref}} \qquad \text{(IV.C.1)}$$

And

$$C_D = \frac{D}{qS_{ref}} \qquad \text{(IV.C.2)}$$

Both coefficients can be assumed to be functions of the following parameters:

$C_L$, $C_D = f$ (Mach number, angel of attack, power on/off, shape) [20]

The Mach number can have a great effect on the coefficients, especially in the transonic and supersonic regime. Of note, at the transonic regime, there is a sudden steep rise in drag before the speed crosses the sonic line, a phenomenon commonly known as transonic drag rise (shown in Figure 19). The main effect, however, is caused by the angle of attack. A slight change can result in significant change in lift coefficient.

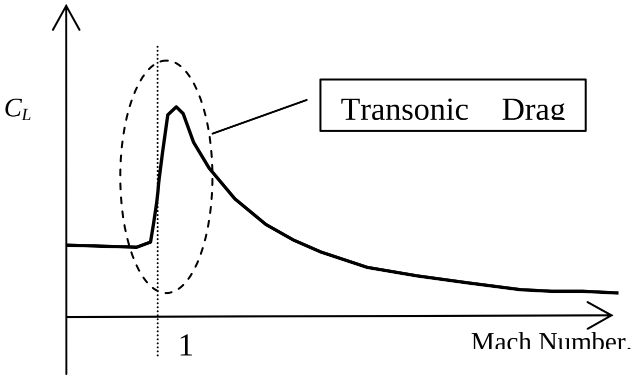


Figure 20. Transonic Drag Rise

When the lift and drag coefficient are plotted against each other, a near parabolic curve emerges, for any given Mach number. The relationship can be expressed as:

$$C_D = C_{D0} + k(C_L - C_{L0})^2 \qquad \text{(IV.C.3)}$$

Graphically, the plot is shown in Figure 19 and is known as the parabolic drag polar:

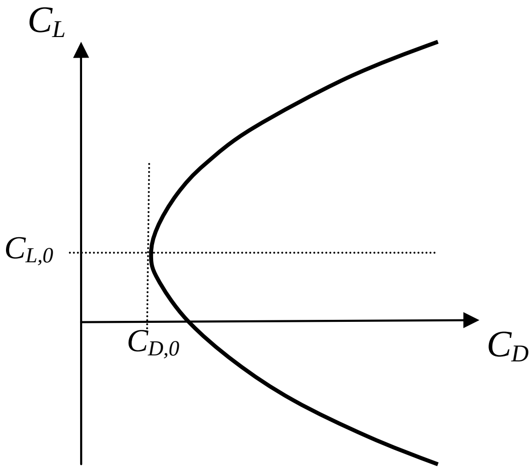


Figure 21. Parabolic Drag Polar for Asymmetric Body

In the drag polar expression, $C_{D0}$ is the zero-lift drag coefficient, a term that accounts for the other drag components when lift is zero while $C_{L0}$ represents the lift coefficient at the lowest drag. The coefficient k is referred to as the induced drag coefficient.

If minimum drag occurs at zero lift, then $C_{L0}$ is zero and the parabola is symmetrical about the $C_D$ axis and centered at $C_{L0} = 0$. This case is representative of a missile, which is axis-symmetric in nature. The different points on the drag polar represent the different angles of attack, which give rise to different lift and drag coefficient. The parabolic drag polar of a missile is as shown in Figure 20:

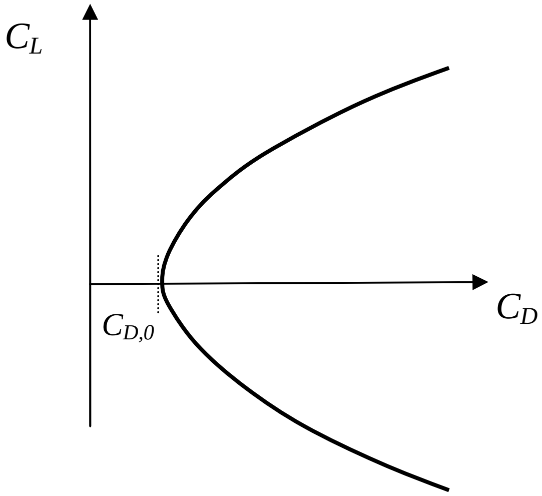


Figure 22. Parabolic Drag Polar for Symmetric Body

In order to implement the induced drag model in the algorithm to calculate drag (*D*), the set of $C_L$-$C_D$ data for the interceptor and target at various Mach number was obtained from open sources and set up in the form of a look-up table. The data is then curve fitted using a polynomial (parabolic) function. A plot of the family of curves for different Mach number using Excel plot is shown in Figure 21:

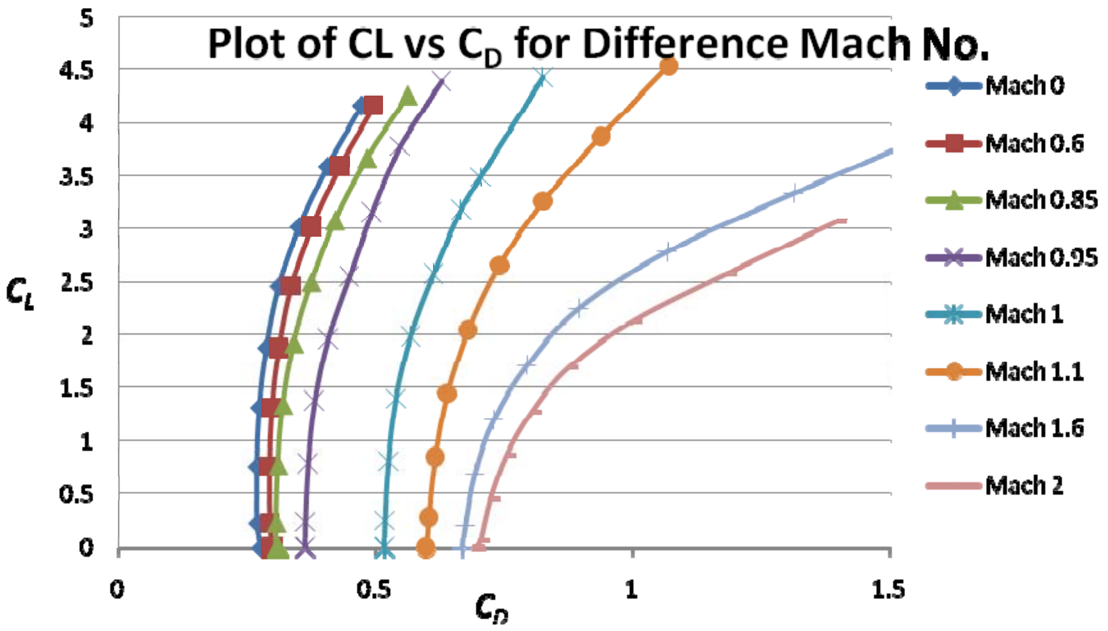


Figure 23. Parabolic Drag Polar for Different Mach Number (Symmetric Body)

The values of $C_{D0}$ and k can be derived from the coefficient of the polynomial function. These values can be substituted into equation (IV.C.3). Since $C_{L0}$ is zero for an axis-symmetric missile, the only outstanding variable that has to be computed is the lift coefficient, $C_L$. For a particular iteration, $C_L$ can be expressed as a function of load factor, $n_z$. The load factor of a missile is defined as:

$$n_z = \frac{L}{W} \qquad \text{(IV.C.4)}$$

Where $L$ is the lift and $W$ is the weight of the missile. From equation (IV.C.1), we can express $L$ in terms of $C_L$:

$$L = C_L q S_{ref} \qquad \text{(IV.C.5)}$$

Therefore, substituting equations (IV.C.5) into (IV.C.4), $C_L$ can be computed:

$$C_L = \frac{2mg}{\rho V^2 S_{ref}} \qquad \text{(IV.C.6)}$$

The overall drag coefficient can then be calculated using equation (IV.C.3), with values of $C_{D0,}$ $k$ and $C_{L.}$

A MATLAB program was developed to implement the induced drag model in the TS guidance algorithm. The code is appended in Appendix C. Minor modifications have to be made in the other sub-routines in order to integrate the new drag model into the algorithm. It is intended to replace the generic drag model currently. Though the new model has been integrated and initially tested to be working, results has not been consistent. Further testing using the new code and more simulation scenario to fine-tune the program can be conducted in future studies.

# V. RESULTS AND DISCUSSIONS

A total of 270 simulations were conducted in this trade-off study. The main objectives of the simulation study are: (1) to verify that the TS guidance can be used in a hit-to-kill interceptor missile to effectively engage a ballistic missile in the boost phase, (2) to predict the region of the interceptor launch position in relations to the ballistic missile launch point which can result in a feasible intercept solution and (3) the impact on the time-to-intercept, altitude and impact angle deviation by varying the intercept geometry coefficient, $W_{IA}$ of the cost function. In this study, simulations were run for various interceptor missile (simulated by the surface-launch SM-6 missile) launch positions, for different northing and easting against a ballistic missile (simulated by the TPD-2 ICBM) launched from a fixed site (arbitrarily fixed at point [0,0,Re] in the earth coordinate system, whereby Re is the radius of the earth). The ballistic missile is launched towards a target in the north direction.

## A. FEASIBILITY OF THE TRAJECTORY SHAPING GUIDANCE

The simulation results showed that the trajectory guidance algorithm worked well for the intercept scenario ran and are within expectation. The guidance provides feasible solutions for the interception of the TPD-2 ballistic missile by the surface launched SM-6 missile. Out of the 270 simulation runs, successful intercept was achieved for 180 runs. The remaining 'non-feasible' intercepts were due to the inability of the guidance algorithm to converge to an optimized problem within the maximum number of iterations allowed. The maximum iterations condition was set to act as the upper bound and 'time-out' condition, reflection the real world situation whereby the intercept scenario does not offer a solution fast enough to ensure a successful intercept. This is realistic as the interceptor missile has to compute the optimized flight path after launch and be guided to the predicted intercept point within a very small time window. Such scenario occurs when the interceptor system is 'out of range' or within the 'blind range' to effectively intercept the ballistic missile within the timeframe of 60 to 180 seconds after launch. Table 22 to 24 presents a summary of simulation results, showing the number of iterations the algorithm takes to find the optimized intercept flight path for values of $W_{IA}$

set to 0, 100 and 1000 respectively. The region in green represents feasible intercepts predicted by the TS algorithm and the numbers refers to the number of iterations required to find the optimized intercept flight path. The letter 'EX' in the red region represented the non-feasible region whereby there is no feasible intercept solution if the interceptor missile is launched from those positions.

| | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 140 | EX | EX | EX | EX | 38 | 58 | 40 | 47 |
| 120 | EX | EX | 64 | 55 | 40 | 36 | 42 | 30 |
| 100 | EX | EX | 44 | 51 | 53 | 40 | 32 | 37 |
| 80 | EX | EX | 43 | 41 | 37 | 26 | 32 | 26 |
| 60 | EX | 45 | 40 | 40 | 26 | 24 | 13 | 12 |
| 40 | EX | 40 | 33 | 32 | 21 | 22 | 10 | EX |
| 20 | EX | 40 | 52 | 26 | 27 | 12 | EX | EX |
| 0 | EX | 57 | 28 | 40 | 16 | 12 | EX | EX |
| -20 | EX | EX | 54 | 27 | 29 | 21 | 21 | 12 |
| -40 | EX | EX | 79 | 50 | 41 | 30 | 22 | 29 |
| -60 | EX | EX | EX | 57 | 40 | 39 | 32 | 40 |
| -80 | EX | EX | EX | EX | EX | 44 | 54 | 41 |

Table 4. No of Iterations Required for Feasible Interceptor Solution (WIA = 0)

| | Westing, km | | | | | |
|---|---|---|---|---|---|---|
| Northing, km | 100 | 80 | 60 | 40 | 20 | 0 |
| 100 | 44 | 51 | 53 | 40 | 32 | 37 |
| 80 | 43 | 41 | 37 | 26 | 32 | 26 |
| 60 | 40 | 40 | 26 | 24 | 13 | 12 |
| 40 | 33 | 32 | 21 | 22 | 10 | Ex |
| 20 | 52 | 26 | 27 | 12 | Ex | Ex |
| 0 | 28 | 40 | 16 | 12 | Ex | Ex |
| -20 | 54 | 27 | 29 | 21 | 21 | 12 |
| -40 | 79 | 50 | 41 | 30 | 12 | 29 |
| -60 | Ex | 57 | 40 | 39 | 32 | 40 |

Table 5. No of Iterations Required for Feasible Interceptor Solution (WIA = 100)

| | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 200 | EX | EX | EX | EX | EX | EX | EX | EX |
| 180 | EX | EX | EX | EX | EX | EX | EX | EX |
| 160 | EX | EX | EX | EX | EX | EX | 90 | 56 |
| 140 | EX | EX | EX | EX | 151 | 165 | 111 | 63 |
| 120 | EX | EX | 64 | 95 | 88 | 120 | 127 | 58 |
| 100 | EX | EX | 123 | 112 | 121 | 182 | 104 | 54 |
| 80 | EX | EX | 128 | 158 | 162 | 150 | 102 | 100 |
| 60 | EX | 114 | 117 | 105 | 115 | 90 | 95 | 92 |
| 40 | EX | 105 | 110 | 113 | 103 | 118 | 95 | EX |
| 20 | EX | 87 | 97 | 81 | 88 | 98 | EX | EX |
| 0 | EX | 56 | 74 | 81 | 79 | 81 | EX | EX |
| -20 | EX | EX | 67 | 96 | 110 | 132 | 107 | 12 |
| -40 | EX | EX | EX | 79 | 106 | 132 | 97 | 26 |
| -60 | EX | EX | EX | 84 | 123 | 198 | 32 | 40 |
| -80 | EX | EX | EX | EX | EX | EX | EX | EX |
| -100 | EX | EX | EX | EX | EX | EX | EX | EX |

Table 6. No of Iterations Required for Feasible Interceptor Solution ($W_{IA}$ = 1000)

The results obtained successfully verified that the TS guidance algorithm does provide feasible solution to the boost phase intercept problem. The new guidance is suitable for scenario involving a high speed hit-to-kill missile interceptor and a fixed trajectory ballistic missile target. By comparing the results for the different intercept geometry coefficient values, it is observed that the number of iterations is increased across the board when $W_{IA}$ is increased from 0 to 1000. This means that when a higher weightage is placed on intercept geometry in the intercept solution, it increases the computing resources required and the optimization process takes a longer time to converge.

## B. REGION OF POSSIBLE DEPLOYMENT POSITION OF THE INTERCEPTOR MISSILE

The simulation study was able to provide a mapping of the feasible intercept region, in terms of the interceptor missile launch position with respect to that of the ballistic missile. As the scenario is symmetrical about the north-south direction, only half of the 360-degree coverage needs to be tested. The other half is identical and a mirror image of the results obtained. The simulation results obtained and plot is appended in Appendix 1. Figure 22 shows the feasible intercept region in green and non-feasible region in red, for the scenario tested. The mapping was done by running simulations and collecting intercept data at intervals of 20 km in the northing and easting direction.

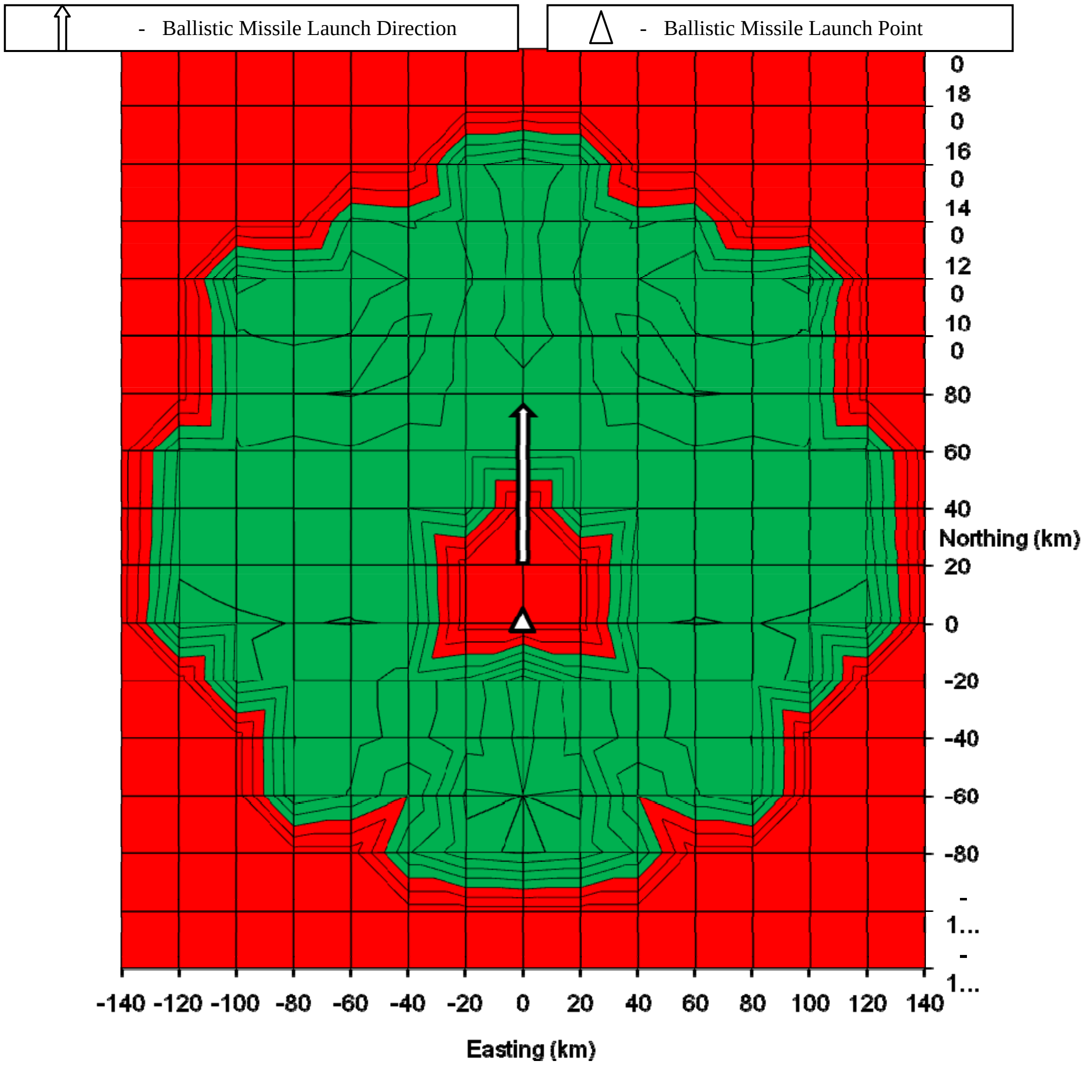


Figure 24. Region of Interceptor Position for Interception of Ballistic Missile

It should be noted that the shape of the boundary has no special significance except for the fact it is purely a 2-D plotting limitation of the software used. A smooth curve should be the correct representation for the boundary of the respective regions. From the mapping, several interesting observations were noted. As expected, the plot shows an off-set in the feasible intercept region with respect to the ballistic missile launch point. This represents a distinct difference in the intercept range between front- or rear-quarter engagement scenarios. One would expect the rear-sector engagement, represented by the intercept launch points south of ballistic missile launch point at [0,0], to have a feasible region with shorter ranges. This is consistent with a typical ‘tail-chase’ intercept scenario, which requires the interceptor to be closer to the target. The front sector engagement is characterized by longer ranges as depicted by the larger area in the plot.

There is a central non-feasible region (in red) and for near ranges close to the ballistic missile launch point. This indicated that certain aerodynamic limits were exceeded, like the g- and lateral acceleration limits, which resulted in a high penalty function value and divergence during the optimization process. The latter represents some kind of ‘blind-range’ whereby intercept is not feasible due to the target being too close for the interceptor to effectively maneuver and to achieve an intercept.

The usefulness of mapping the region of feasible intercept is that it provides critical information for the operational planners on the possible region to deploy an interceptor system (using the TS guidance) given a ballistic missile threat. The creation of such plots can be automated and done in a short time using computer program. The input requirements are the specifications of the interceptor missile and intelligence on the specifications/characteristics of the target missile. The accuracy of such prediction software would depend on the number of simulation run (proportional to the ‘resolution’ or distance interval required) and accuracy of the input data and model used. There is scope to develop a computer program to generate intercept region plots for operational planning purposes in future studies.

## C. EFFECT OF VARYING THE INTERCEPT GEOMETRY COEFFICIENT

One of the main objectives of the trajectory-shaping guidance is to effect maximum transfer of kinetic energy from the interceptor missile to the target missile in a hit-to-kill interception scenario. This can be achieved when the impact angle onto the target missile is at a right angle to the ballistic missile body. The TS guidance algorithm allows the missile designer to adjust the weightage placed on the intercept geometry, amongst other critical parameters, by varying the value of $W_{IA}$, which is to be optimized (minimized). The trade-off study examines the effect on time-to-go, intercept altitude and impact angle deviation due to changes in $W_{IA.}$

In this study, simulations were run for three values of $W_{IA}$, namely 0, 100 and 1000. By having $W_{IA} = 0$ implies that the optimized flight path does not take into account the end-game intercept geometry. In the case of $W_{IA} = 1000$, a higher weightage is placed on impact angle at intercept and the flight path will be optimized to achieve an impact angle as close to 90 degrees as possible. When a higher ‘premium’ is placed on the intercept geometry, the algorithm will attempt it at the expense of other performance parameter. Here, there is always some form of trade-off that a missile designer has to be conscious of.

A comparison of the time-to-go or time-to-intercept, intercept altitude and impact angle deviation for different values of $W_{IA}$ provides valuable insights into the effects of $W_{IA}$ on the missile performance under the same intercept scenario. Table 7 presents a summary of the comparison results.

| WIA | Time-to-Intercept (s) | | | Intercept Altitude (km) | | | Impact Angle Dev (°) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Min | Max | Ave | Min | Max | Ave | Min | Max | Ave |
| 0 | 17.6 | 61.0 | 38.67 | 22.3 | 60.0 | 38.77 | 2.4 | 30.3 | 18.45 |
| 100 | 17.6 | 61.0 | 38.48 | 22.3 | 60.0 | 38.77 | 2.4 | 30.3 | 18.45 |
| 1000 | 17.8 | 60.2 | 39.00 | 22.4 | 59.1 | 38.94 | 0.0 | 30.0 | 10.02 |

Table 7. Time-To-Intercept, Altitude And Impact Angle Deviation For Different Intercept Geometry Coefficient, $W_{IA}$

Generally, it can be seen that there is no significant different between the results obtained for $W_{IA}$ being set at 0 and 100, for the time-to-intercept, intercept altitude and impact angle deviation. However, some different is observed when the weightage coefficient is increased by a factor of 1000. The most significant difference is observed in the impact angle deviation. With a high weightage placed on intercept geometry ($W_{IA}$ = 1000), the minimum impact angle deviation is 0, and the average over all the feasible intercepts reduces from about 18 ° to 10°· This is within expectation and verified that the algorithm indeed attempted to minimize the impact angle deviation when the $W_{IA}$ is increased. Another important observation is that by enhancing the intercept geometry weightage, it does not affect time-to-intercept and intercept altitude. By correlating the observations made in the previous section, it can be seen that when $W_{IA}$ is increased, the number of iterations needed for convergence also goes up. However, the overall time taken for the intercept generally does not increase significantly. This seems possible if there is some trade-off or interaction with the other parameter(s) which is not investigated in this study. It can also be seen that the area of intercept decreases when $W_{IA}$ is changed from 0 to 1000. A possible reason is that the larger number of iterations required for higher intercept geometry coefficient may have resulted in the further range intercepts exceeding the maximum iteration limit imposed by the guidance. This will cause the longer range region to be marked as non-feasible. However, this clearly shows the 'penalty' for imposing a placing a higher weightage on intercept geometry. Figure 23 provides an illustration of the typical intercept geometry at $W_{IA}$ = 1000. It can be seen that the flight path is optimized for a near $90^o$ impact to the ballistic missile trajectory.

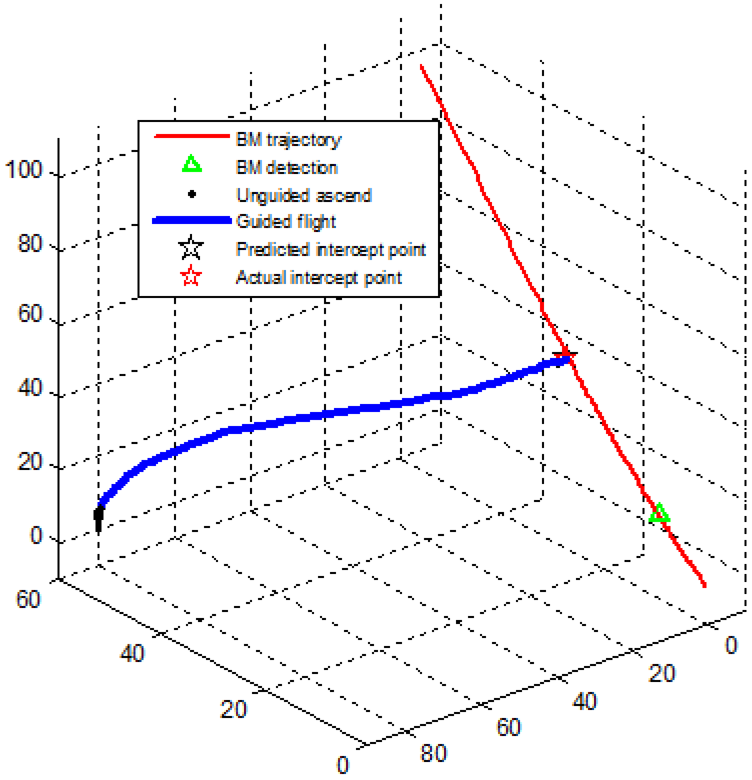


Figure 25. Typical Intercept Geometry

It can be expected that when the weightage for intercept geometry is increased, there is greater constraints imposed on the interceptor system to optimized impact angle amongst other factors. This will affect missile performance in terms of time-to-go and higher intercept altitude. A quick sensitivity study by using WIA values of 0, 100 and 1000 and computing the statistical average of all feasible intercept for time-to-go, intercept altitude and impact angle deviation in the study revealed that increasing the weightage by a factor of 100 does not result in significant changes in the missile performance. At a factor of 1000, the effect on the missile flight parameters begins to be more observable. More simulations can be conducted in future studies for higher values of the $W_{IA}$ that will cause significant effect on missile performance. Some form of scaling factor can be obtained, from which a missile designer can select suitable values $W_{IA}$ to use for different missions.

There is hence a trade-off between the weightage coefficient and other performance parameters of the interceptor missile. From the trade-off study, it was demonstrated that the TS guidance allows the missile designer the flexibility to adjust the weightage of the critical parameters in order to optimize the intercept path. With the insight obtained from the study, the effect of the weightage coefficient on missile performance is known. There is a need to balance the trade-off between the weightage placed on the different critical parameters. In the particular case of intercept geometry, the weightage coefficient ($W_{IA}$) must be carefully selected to optimize missile performance for a particular mission or scenario. The same consideration can be translated to the other critical parameters. Again, a computer program can be developed to aid missile designer in selecting a suitable weightage or cost function coefficients for different mission requirement and operational scenario. This will be very useful if TS guidance in implemented in intercept missiles and offers greater advantage over other guidance laws as one that allows 'customization' for different mission requirements simply using software changes.

## D. INTERCEPT GEOMETRY TRENDS

The simulation study also provides good data to perform simple trending analysis. Useful charts can be generated to provide trending information. An example is the impact angle deviation data derived from the simulation. It provides greater insights into how the intercept geometry will change with respect the ballistic missile trajectory azimuth angle,

$\theta_{BM}$, with respect to the interceptor launch site. The angle $\theta_{BM,}$ is defined as the angle between the trajectory plane of the ballistic missile and the direction from the interceptor launch point to the detected ballistic missile position, as shown in Figure 26.

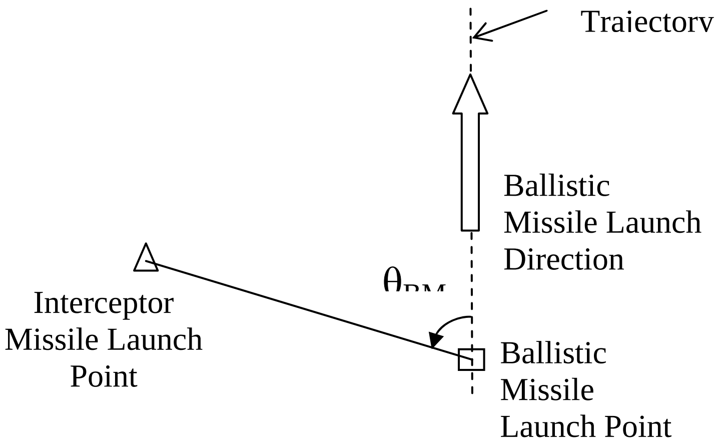


Figure 26. Illustration of Angle between interceptor Launch Position and Ballistic Missile Trajectory Plane

Figure 27 shows the plot of impact angle deviation for interceptors launched from the various positions.

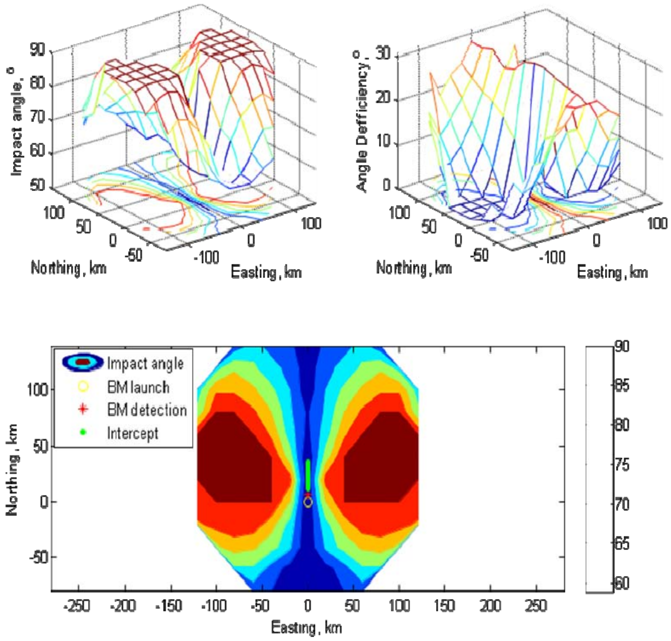


Figure 27. Effect of Interceptor Launch Direction on Intercept Geometry

It can be observed from the plot that interceptors launched normal to the ballistic missile trajectory plane ($\theta_{BM}$ = $90^o$) usually have small impact angle deviation (very close to $0^o$) or impact the ballistic missile at close to $90^{o.}$ The impact angle deviation increases when the interceptor flight path becomes more oblique to the ballistic missile trajectory plane (that is, $\theta_{BM}$ becomes smaller). For example, the impact angle deviation is larger, when the interceptor is fired from [80 km N, 0 km E], as compared to an interceptor missile being launched from [0 km N, -50km E]

The largest impact angle deviation occurs when the interceptor missile is in the front-sector of the ballistic missile. This can be explained from the fact that the missile does not have sufficient time to effectively 'shape' its flight path for a 90o impact on the ballistic missile, compared to the cases where the interceptor is closing-in from the side (normal to the ballistic missile trajectory plane). By plotting the results for the other parameters and analyzing them in future studies, other useful insight into the interceptor missile performance can be deduced. This will be helpful to missile designers as well as operational planners to optimize the potential of the TS guidance for possible anti-ballistic missile defense missions.

# VI. CONCLUSIONS

The simulation study verified that the TS guidance provides feasible solutions to the boost phase intercept problem involving the interception of a fixed-trajectory ballistic missile by a surface-launched interceptor missile. For the particular scenario used in the study, it was shown that if an interceptor system using the TS guidance can be deployed within 30 to 60 km (for front sector intercept) and 20 – 50 km for (rear-sector intercept), it is possible to intercept the ballistic missile effectively during the boost phase. The results also shows that the new guidance allows missile designer to ‘customize’ the guidance algorithm for specific mission by changing the ‘weights’ of some critical parameters However, there is a trade-off between the each of the weightage coefficient and missile performance. By enhancing the weightage on intercept geometry, there is no change in the time-to-intercept or intercept altitude. However, the region for the feasible interceptor launch position around the ballistic missile launch point has reduced. Hence, the weightage coefficient will have to be carefully chosen to ensure that there is a balance of benefit and penalty with respect to specific missions. This aspect of ‘customization’ represents a clear advantage of the TS guidance over other guidance law. The new guidance algorithm can be further enhanced and fine-tuned. Further research can include conducting simulation study using a 6 DoF model of the TS guidance, combining guidance solution with on-board navigation solution, testing the complete guidance, navigation and control solutions and developing a computer program as a design tool to perform trade-off study and select weightage coefficients for different mission requirements. The new TS guidance holds promise to be a feasible and implementable solution to the boost phase ballistic missile intercept problem that will become more prominent in the coming years.

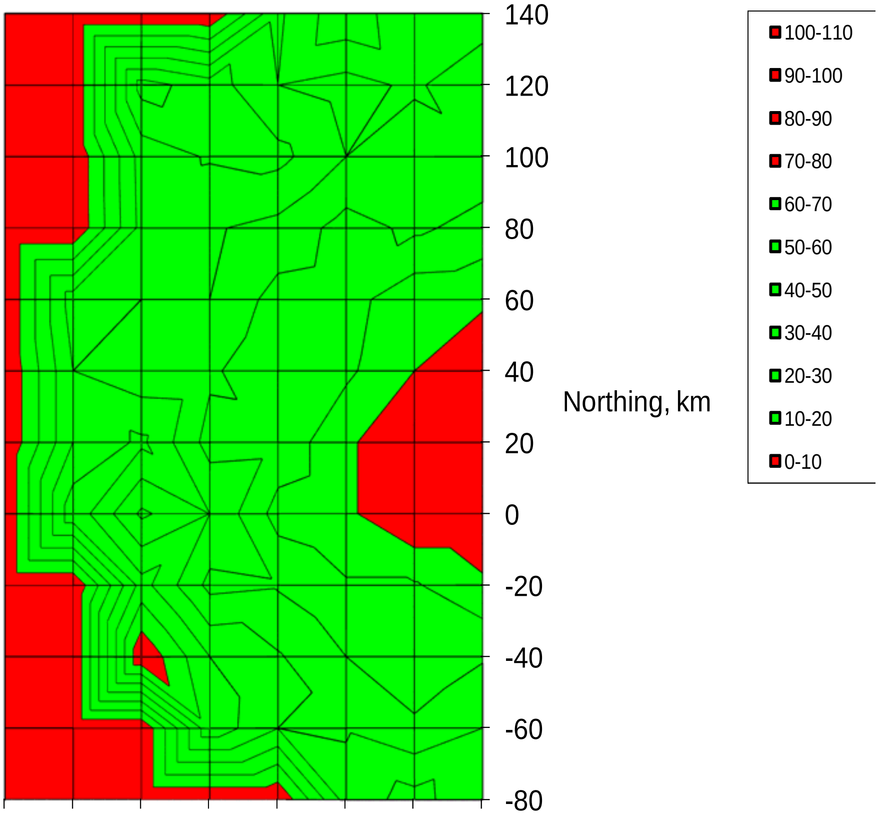
THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX A: TABLES AND PLOTS OF SIMULATION RESULTS

NO OF ITERATIONS WIA = 0

| Northing, km | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 140 | EX | EX | EX | EX | 38 | 58 | 40 | 47 |
| 120 | EX | EX | 64 | 55 | 40 | 36 | 42 | 30 |
| 100 | EX | EX | 44 | 51 | 53 | 40 | 32 | 37 |
| 80 | EX | EX | 43 | 41 | 37 | 26 | 32 | 26 |
| 60 | EX | 45 | 40 | 40 | 26 | 24 | 13 | 12 |
| 40 | EX | 40 | 33 | 32 | 21 | 22 | 10 | EX |
| 20 | EX | 40 | 52 | 26 | 27 | 12 | EX | EX |
| 0 | EX | 57 | 28 | 40 | 16 | 12 | EX | EX |
| -20 | EX | EX | 54 | 27 | 29 | 21 | 21 | 12 |
| -40 | EX | EX | 79 | 50 | 41 | 30 | 22 | 29 |
| -60 | EX | EX | EX | 57 | 40 | 39 | 32 | 40 |
| -80 | EX | EX | EX | EX | EX | 44 | 54 | 41 |

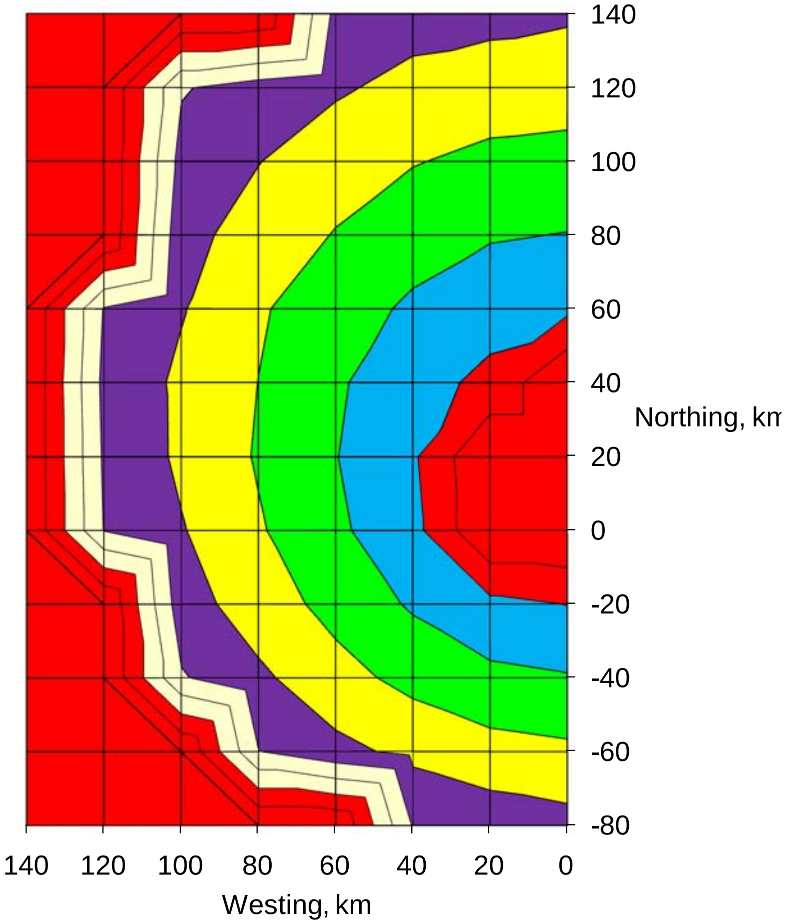
**Ex - Exceed Max No. of Iterations - Non-feasible Intercept**



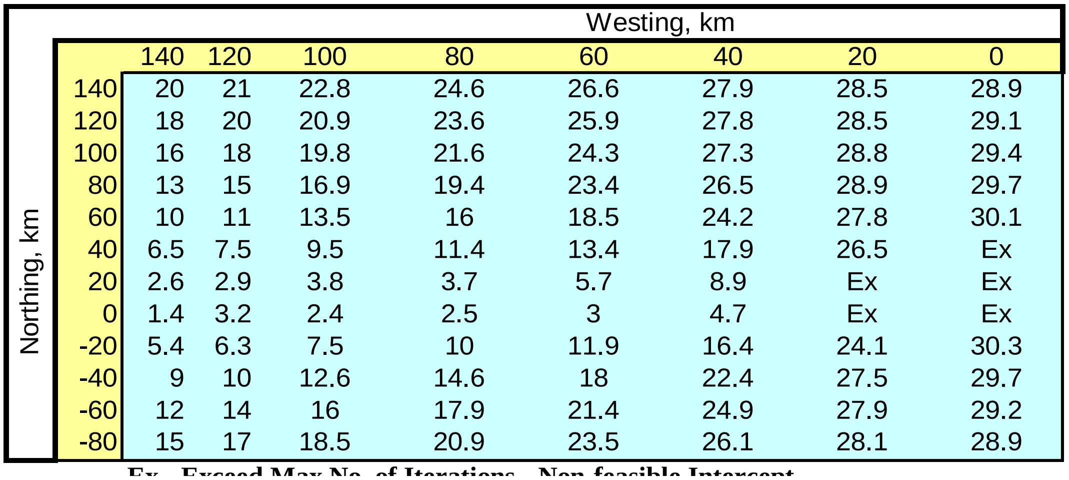
TIME-TO-GO WIA = 0

| | Westing, km | | | | | | |
|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 |
| 140 | 100 | 100 | 100 | 100 | 57 | 53.9 | 52.9 |
| 120 | 100 | 100 | 60.9 | 55.2 | 51.3 | 47.1 | 44.8 |
| 100 | 100 | 100 | 56.4 | 50.3 | 44.5 | 40.5 | 37.8 |
| 80 | 100 | 100 | 53.1 | 45.9 | 39.5 | 34.2 | 30.8 |
| 60 | 100 | 59 | 50.7 | 41.07 | 34.5 | 28.4 | 23.9 |
| 40 | 100 | 58 | 48.2 | 39.9 | 31.3 | 23.8 | 17.6 |
| 20 | 100 | 59 | 48.3 | 39.1 | 30.3 | 21.4 | 0 |
| 0 | 100 | 60 | 50.7 | 41 | 31.8 | 23.1 | 0 |
| -20 | 100 | 100 | 54.3 | 45 | 36.8 | 28.8 | 22.5 |
| -40 | 100 | 100 | 61 | 51.9 | 43.5 | 37 | 32.3 |
| -60 | 100 | 100 | 100 | 60.2 | 52.7 | 47.6 | 43.6 |
| -80 | 100 | 100 | 100 | 100 | 100 | 59.2 | 55.8 |

**Note : Value of 0 and 100 Assigned for Non-Feasible Intercept**

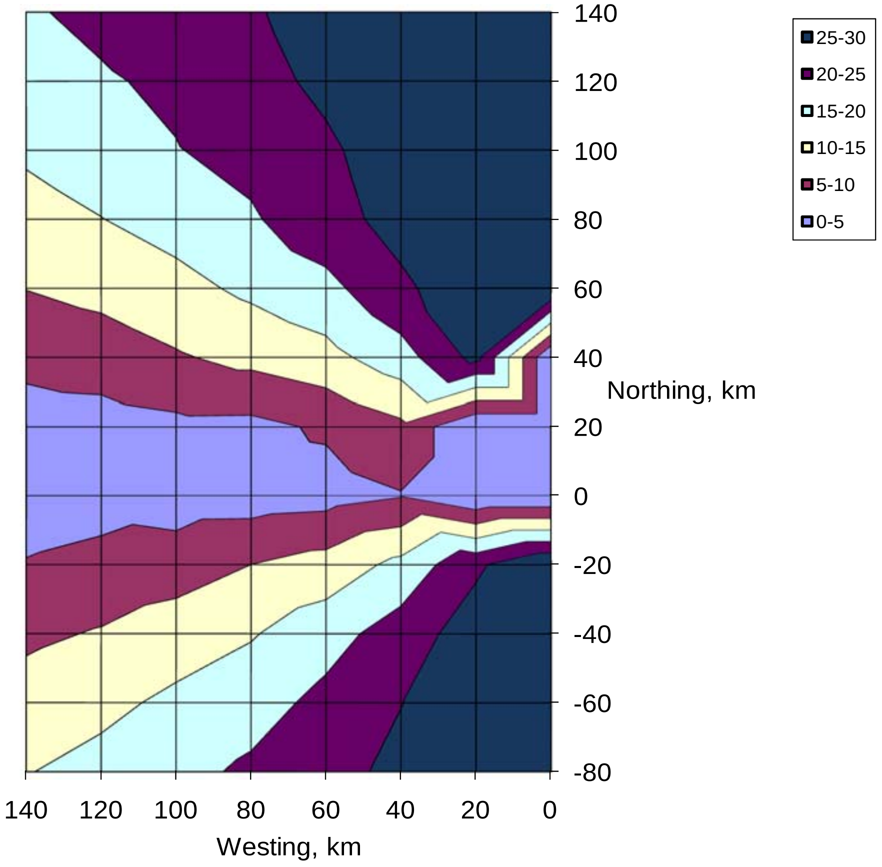


IMPACT ANGLE WIA = 0



| | | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| Northing, km | 140 | 20 | 21 | 22.8 | 24.6 | 26.6 | 27.9 | 28.5 | 28.9 |
| | 120 | 18 | 20 | 20.9 | 23.6 | 25.9 | 27.8 | 28.5 | 29.1 |
| | 100 | 16 | 18 | 19.8 | 21.6 | 24.3 | 27.3 | 28.8 | 29.4 |
| | 80 | 13 | 15 | 16.9 | 19.4 | 23.4 | 26.5 | 28.9 | 29.7 |
| | 60 | 10 | 11 | 13.5 | 16 | 18.5 | 24.2 | 27.8 | 30.1 |
| | 40 | 6.5 | 7.5 | 9.5 | 11.4 | 13.4 | 17.9 | 26.5 | Ex |
| | 20 | 2.6 | 2.9 | 3.8 | 3.7 | 5.7 | 8.9 | Ex | Ex |
| | 0 | 1.4 | 3.2 | 2.4 | 2.5 | 3 | 4.7 | Ex | Ex |
| | -20 | 5.4 | 6.3 | 7.5 | 10 | 11.9 | 16.4 | 24.1 | 30.3 |
| | -40 | 9 | 10 | 12.6 | 14.6 | 18 | 22.4 | 27.5 | 29.7 |
| | -60 | 12 | 14 | 16 | 17.9 | 21.4 | 24.9 | 27.9 | 29.2 |
| | -80 | 15 | 17 | 18.5 | 20.9 | 23.5 | 26.1 | 28.1 | 28.9 |

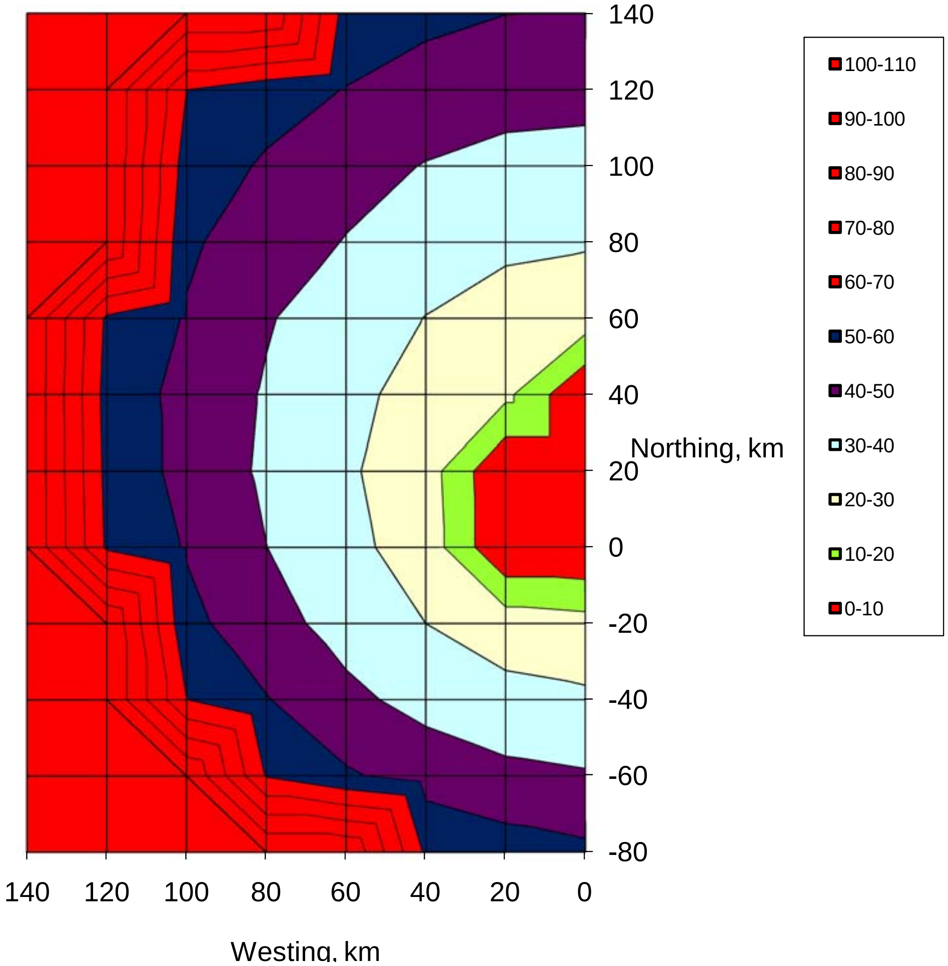
Ex - Exceed Max No. of Iterations - Non feasible Intercept



ALTITUDE WIA = 0

| | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 140 | 100 | 100 | 100 | 100 | 55.7 | 52.4 | 50.1 | 49.7 |
| 120 | 100 | 100 | 59.9 | 53.9 | 49.7 | 45.8 | 43.6 | 43 |
| 100 | 100 | 100 | 55 | 48.9 | 43.3 | 39.6 | 37.2 | 36.6 |
| 80 | 100 | 100 | 51.5 | 44.7 | 39.6 | 34.4 | 31.6 | 30.7 |
| 60 | 100 | 58 | 49.3 | 40.8 | 34.5 | 29.8 | 26.5 | 25.2 |
| 40 | 100 | 56 | 46.8 | 39.1 | 32.6 | 26.4 | 22.3 | Ex |
| 20 | 100 | 57 | 46.8 | 38.4 | 31.2 | 24.8 | Ex | Ex |
| 0 | 100 | 58 | 49.3 | 40.1 | 32.4 | 25.9 | Ex | Ex |
| -20 | 100 | 100 | 52.7 | 43.7 | 36.3 | 30 | 25.5 | 23.6 |
| -40 | 100 | 100 | 60 | 50.5 | 42.4 | 36.7 | 32.8 | 31.5 |
| -60 | 100 | 100 | 100 | 59.1 | 51.2 | 46.1 | 42.5 | 40.9 |
| -80 | 100 | 100 | 100 | 100 | 100 | 58.1 | 54.5 | 52 |

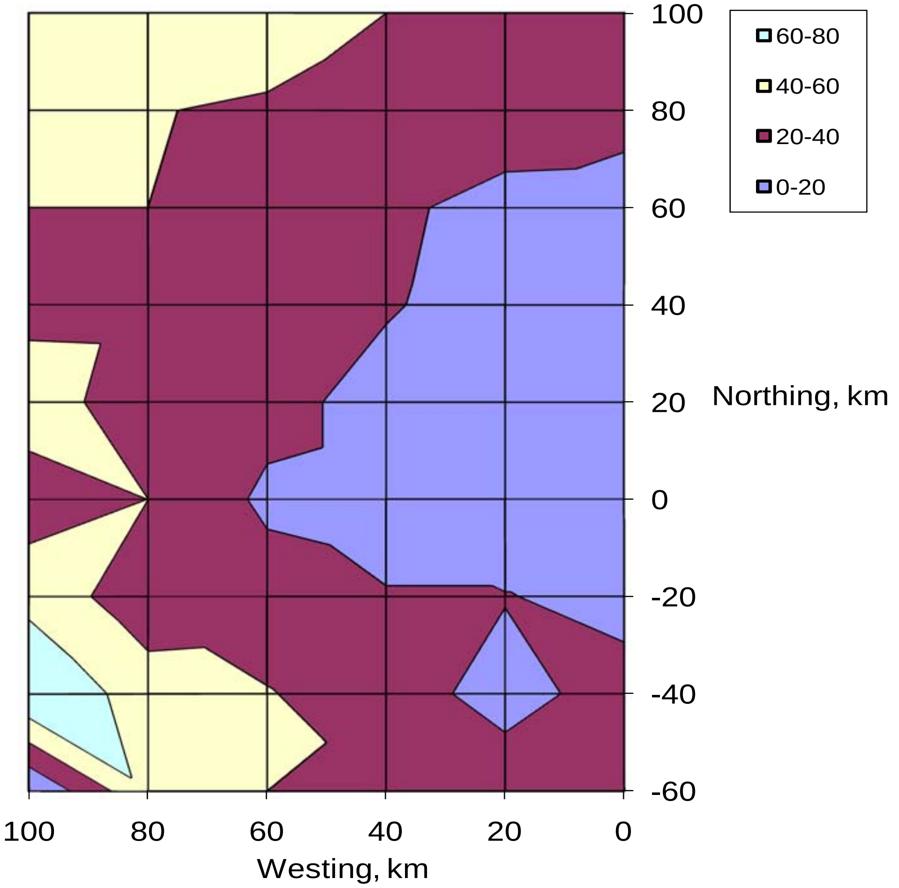
**100 or Ex - Exceed Max No. of Iterations - Non-feasible Intercept**



NO OF ITERATIONS WIA = 100

| | | Westing, km | | | | | |
|---|---|---|---|---|---|---|---|
| | | 100 | 80 | 60 | 40 | 20 | 0 |
| Northing, km | 100 | 44 | 51 | 53 | 40 | 32 | 37 |
| | 80 | 43 | 41 | 37 | 26 | 32 | 26 |
| | 60 | 40 | 40 | 26 | 24 | 13 | 12 |
| | 40 | 33 | 32 | 21 | 22 | 10 | Ex |
| | 20 | 52 | 26 | 27 | 12 | Ex | Ex |
| | 0 | 28 | 40 | 16 | 12 | Ex | Ex |
| | -20 | 54 | 27 | 29 | 21 | 21 | 12 |
| | -40 | 79 | 50 | 41 | 30 | 12 | 29 |
| | -60 | Ex | 57 | 40 | 39 | 32 | 40 |

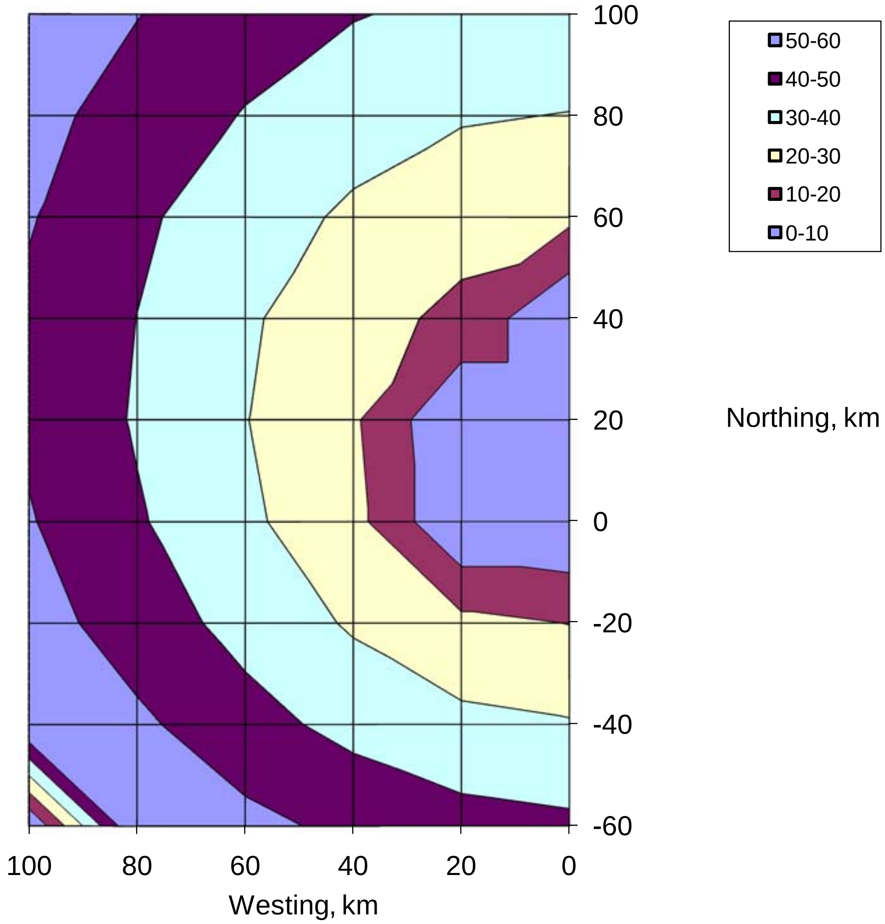
EX - No. of Iterations Exceeded. Represents Non-Feasible Region



TIME-TO-GO $W_{IA}$ = 100

|  |  | Westing, km |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  |  | 100 | 80 | 60 | 40 | 20 | 0 |
| Northing, km | 100 | 56.4 | 50.3 | 44.5 | 40.5 | 37.8 | 36.9 |
|  | 80 | 53.1 | 45.9 | 39.5 | 34.2 | 30.8 | 29.7 |
|  | 60 | 50.7 | 41.7 | 34.5 | 28.4 | 23.9 | 22.1 |
|  | 40 | 48.2 | 39.9 | 31.3 | 23.8 | 17.6 | ex |
|  | 20 | 48.3 | 39.1 | 30.3 | 21.4 | ex | ex |
|  | 0 | 50.7 | 41 | 31.8 | 23.1 | ex | ex |
|  | -20 | 54.3 | 45 | 36.8 | 28.8 | 22.5 | 19.8 |
|  | -40 | 61 | 51.9 | 43.5 | 37 | 32.3 | 30.7 |
|  | -60 | ex | 60.2 | 52.7 | 47.6 | 43.6 | 41.9 |

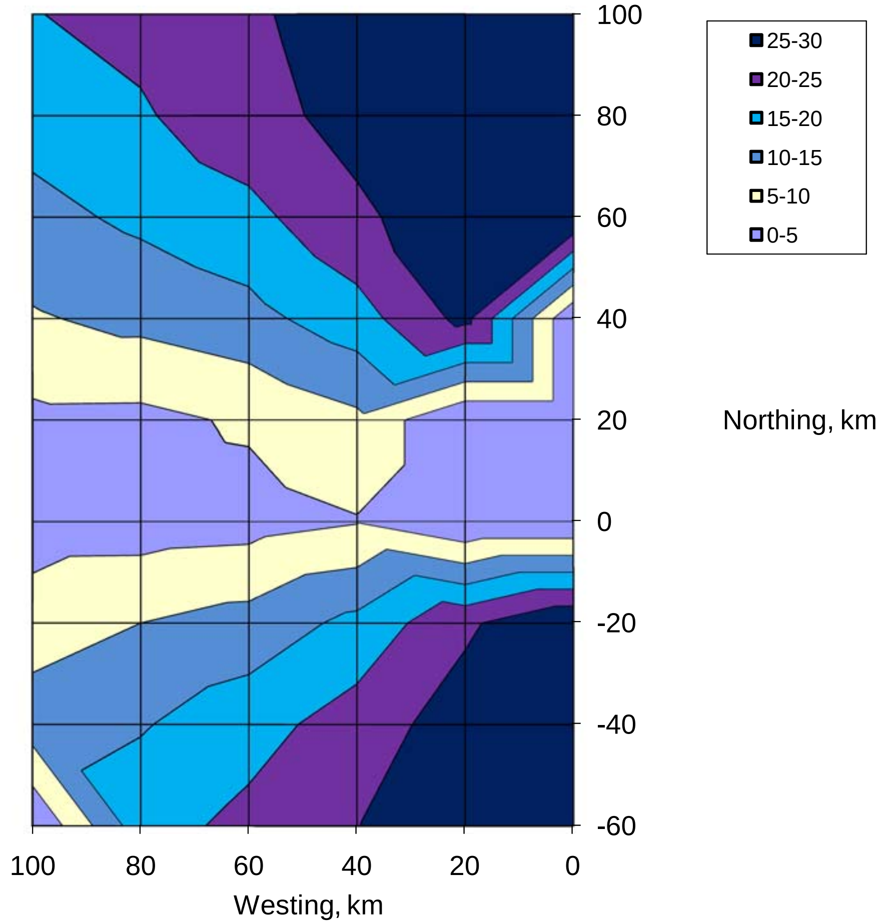
EX - Represents Non-Feasible Region



IMPACT ANGLE DEF $W_{IA} = 100$

| | | Westing, km | | | | | |
|---|---|---|---|---|---|---|---|
| | | 100 | 80 | 60 | 40 | 20 | 0 |
| Northing, km | 100 | 19.8 | 21.6 | 24.3 | 27.3 | 28.8 | 29.4 |
| | 80 | 16.9 | 19.4 | 23.4 | 26.5 | 28.9 | 29.7 |
| | 60 | 13.5 | 16 | 18.5 | 24.2 | 27.8 | 30.1 |
| | 40 | 9.5 | 11.4 | 13.4 | 17.9 | 26.5 | ex |
| | 20 | 3.8 | 3.7 | 5.7 | 8.9 | ex | ex |
| | 0 | 2.4 | 2.5 | 3 | 4.7 | ex | ex |
| | -20 | 7.5 | 10 | 11.9 | 16.4 | 24.1 | 30.3 |
| | -40 | 12.6 | 14.6 | 18 | 22.4 | 27.5 | 29.7 |
| | -60 | EX | 17.9 | 21.4 | 24.9 | 27.9 | 29.2 |

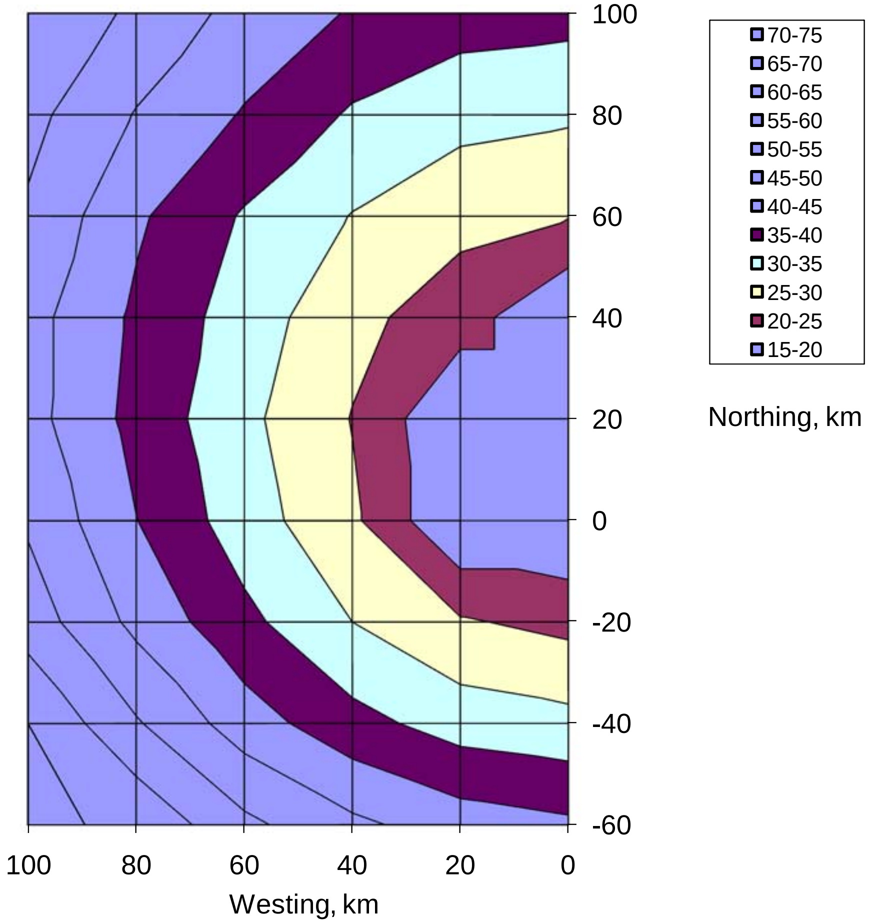
EX - Represents Non-Feasible Region



INTERCEPT ALTITUDE $W_{IA} = 100$

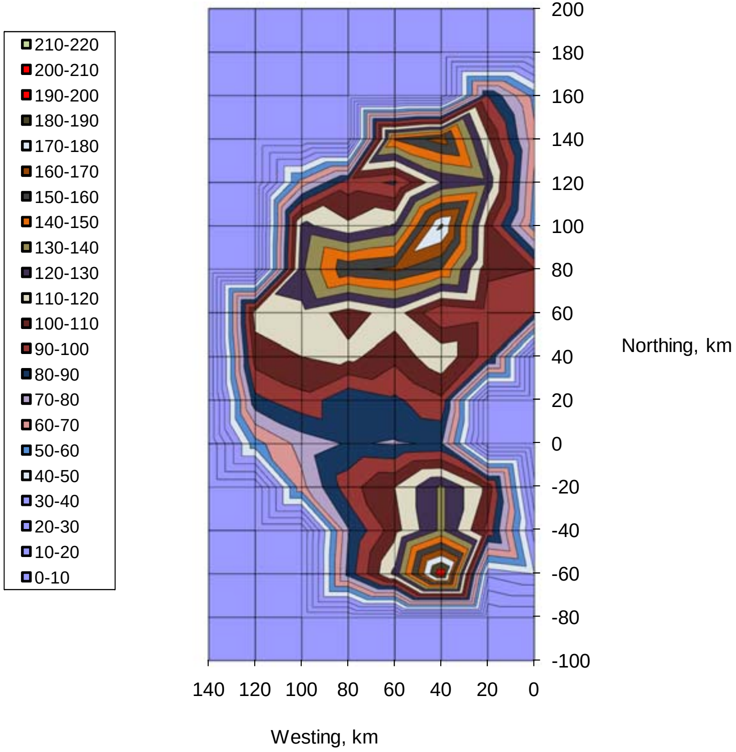
| | | Westing, km | | | | | |
|---|---|---|---|---|---|---|---|
| | | 100 | 80 | 60 | 40 | 20 | 0 |
| Northing, km | 100 | 55 | 48.9 | 43.3 | 39.6 | 37.2 | 36.6 |
| | 80 | 51.5 | 44.7 | 39.6 | 34.4 | 31.6 | 30.7 |
| | 60 | 49.3 | 40.8 | 34.5 | 29.8 | 26.5 | 25.2 |
| | 40 | 46.8 | 39.1 | 32.6 | 26.4 | 22.3 | ex |
| | 20 | 46.8 | 38.4 | 31.2 | 24.8 | ex | ex |
| | 0 | 49.3 | 40.1 | 32.4 | 25.9 | ex | ex |
| | -20 | 52.7 | 43.7 | 36.3 | 30 | 25.5 | 23.6 |
| | -40 | 60 | 50.5 | 42.4 | 36.7 | 32.8 | 31.5 |
| | -60 | 61 | 59.1 | 51.2 | 46.1 | 42.5 | 40.9 |

EX - Represents Non-Feasible Region



NO. OF ITERATIONS WIA = 1000

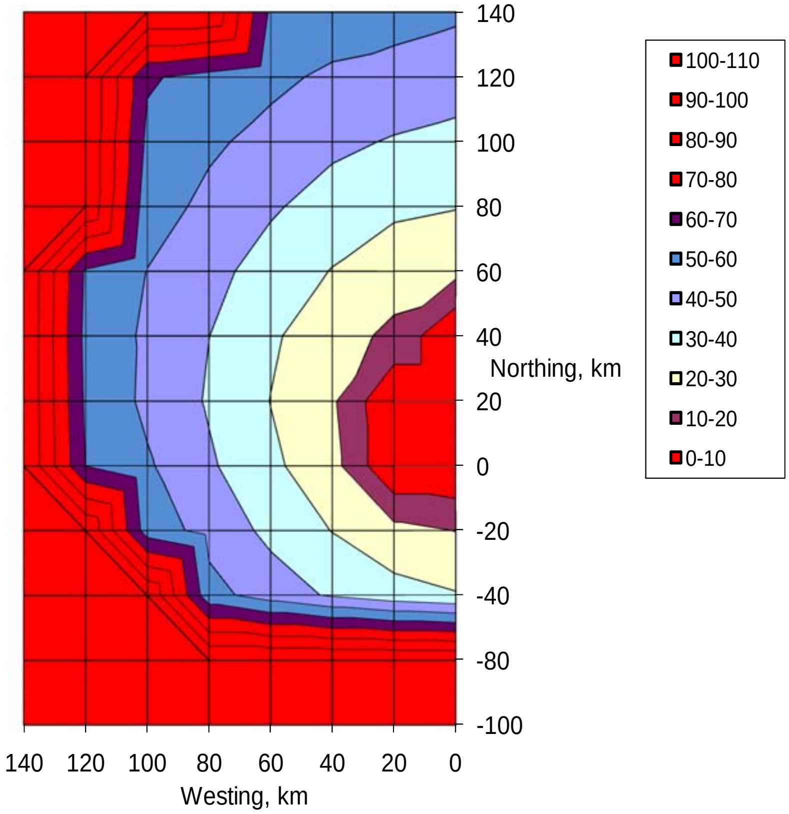
| | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 200 | EX | EX | EX | EX | EX | EX | EX | EX |
| 180 | EX | EX | EX | EX | EX | EX | EX | EX |
| 160 | EX | EX | EX | EX | EX | EX | 90 | 56 |
| 140 | EX | EX | EX | EX | 151 | 165 | 111 | 63 |
| 120 | EX | EX | 64 | 95 | 88 | 120 | 127 | 58 |
| 100 | EX | EX | 123 | 112 | 121 | 182 | 104 | 54 |
| 80 | EX | EX | 128 | 158 | 162 | 150 | 102 | 100 |
| 60 | EX | 114 | 117 | 105 | 115 | 90 | 95 | 92 |
| 40 | EX | 105 | 110 | 113 | 103 | 118 | 95 | EX |
| 20 | EX | 87 | 97 | 81 | 88 | 98 | EX | EX |
| 0 | EX | 56 | 74 | 81 | 79 | 81 | EX | EX |
| -20 | EX | EX | 67 | 96 | 110 | 132 | 107 | 12 |
| -40 | EX | EX | EX | 79 | 106 | 132 | 97 | 26 |
| -60 | EX | EX | EX | 84 | 123 | 198 | 32 | 40 |
| -80 | EX | EX | EX | EX | EX | EX | EX | EX |
| -100 | EX | EX | EX | EX | EX | EX | EX | EX |



TIME-TO-GO $W_{IA} = 1000$

| | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 200 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 180 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 160 | 100 | 100 | 100 | 100 | 100 | 100 | 60 | 59.3 |
| 140 | 100 | 100 | 100 | 100 | 58.2 | 55.6 | 53.3 | 51.5 |
| 120 | 100 | 100 | 61.2 | 56.6 | 52.1 | 48.3 | 46.9 | 44.7 |
| 100 | 100 | 100 | 57.6 | 51.5 | 47.2 | 42.2 | 39.2 | 37.2 |
| 80 | 100 | 100 | 54.2 | 47.8 | 41.3 | 35.7 | 31.8 | 30.4 |
| 60 | 100 | 58.7 | 49.8 | 43.1 | 35.8 | 29.6 | 24.6 | 22.7 |
| 40 | 100 | 57.4 | 48.3 | 40.1 | 31.4 | 24.2 | 17.8 | Ex |
| 20 | 100 | 58.4 | 47.9 | 39.0 | 29.8 | 21.4 | Ex | Ex |
| 0 | 100 | 59.8 | 51.2 | 41.4 | 32.0 | 23.5 | Ex | Ex |
| -20 | 100 | 100 | 55.1 | 46.8 | 37.5 | 29.7 | 22.9 | 19.8 |
| -40 | 100 | 100 | 100 | 53.5 | 45.3 | 38.7 | 33.7 | 30.7 |
| -60 | 100 | 100 | 100 | 60.2 | 54.4 | 47.4 | 43.6 | 41.9 |
| -80 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| -100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

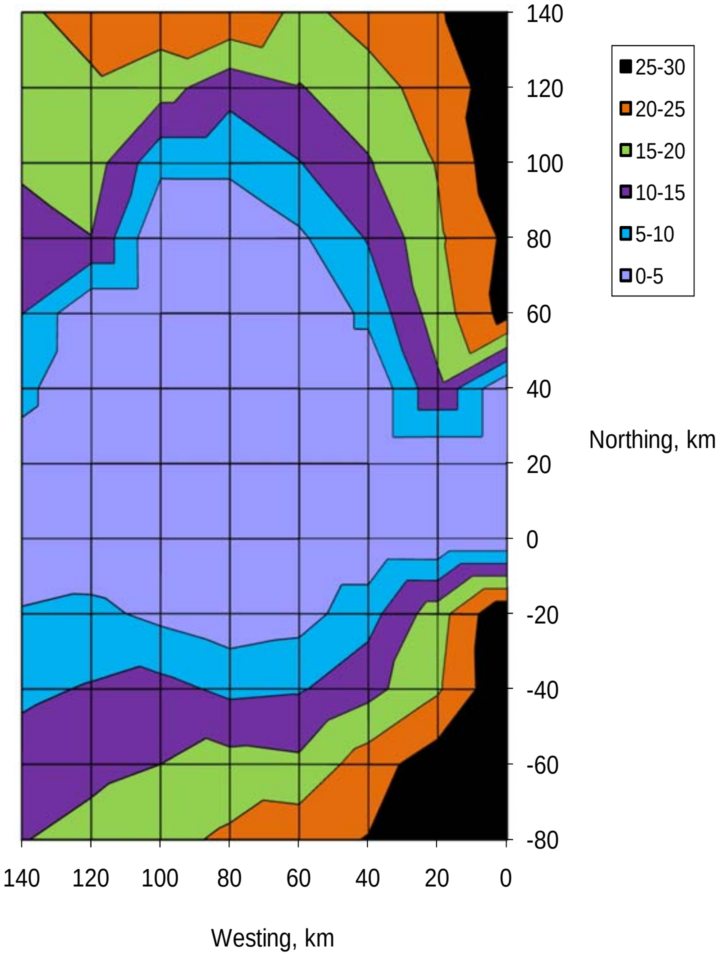
EX - Represents Non-Feasible Region



IMPACT ANGLE DEVIATION $W_{IA}$ = 1000

| | | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| Northing, km | 140 | 20 | 21 | 22.8 | 24.6 | 18.6 | 22 | 24.6 | 28.6 |
| | 120 | 18 | 20 | 17.1 | 11.6 | 14.7 | 18 | 22.2 | 28 |
| | 100 | 16 | 18 | 6.3 | 6.3 | 9.8 | 14.6 | 20.3 | 29.1 |
| | 80 | 13 | 15 | 0 | 0.1 | 4.1 | 10.4 | 19.2 | 26 |
| | 60 | 10 | 0.1 | 0.1 | 0 | 0 | 6.3 | 17.6 | 27 |
| | 40 | 6.5 | 0 | 0.1 | 0 | 0 | 0.1 | 13.9 | ex |
| | 20 | 2.6 | 0 | 0 | 0.1 | 0 | 0.1 | ex | ex |
| | 0 | 1.4 | 1.2 | 0 | 0 | 0.1 | 0.1 | ex | ex |
| | -20 | 5.4 | 6.3 | 3.7 | 1.6 | 2.9 | 8.1 | 17.9 | 30 |
| | -40 | 9 | 10 | 11.6 | 8.9 | 9.5 | 13.3 | 19.3 | 29.7 |
| | -60 | 12 | 14 | 15 | 16.8 | 16 | 22.7 | 27.9 | 29.2 |
| | -80 | 15 | 17 | 18.5 | 20.9 | 23.5 | 25.2 | 28.1 | 28.9 |

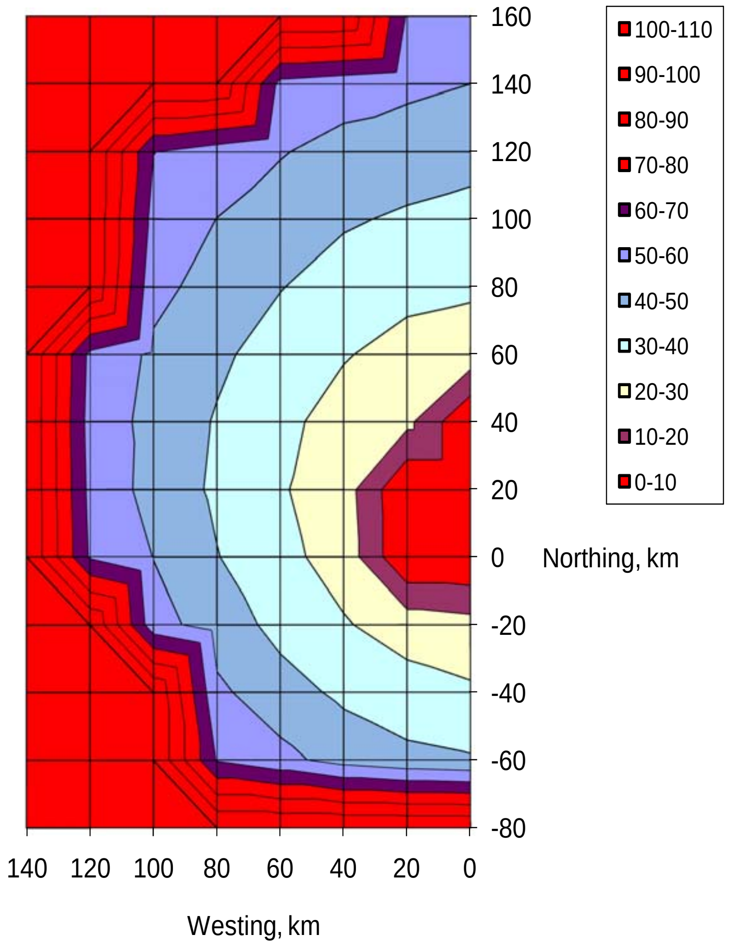
EX - Represents Non-Feasible Region



INTERCEPT ALTITUDE $W_{IA} = 1000$

| | Westing, km | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Northing, km | 140 | 120 | 100 | 80 | 60 | 40 | 20 | 0 |
| 160 | 100 | 100 | 100 | 100 | 100 | 100 | 58.9 | 58 |
| 140 | 100 | 100 | 100 | 100 | 56.9 | 54.3 | 51.9 | 50 |
| 120 | 100 | 100 | 60.2 | 55.2 | 50.6 | 46.9 | 45.5 | 43.5 |
| 100 | 100 | 100 | 56.4 | 49.9 | 45.8 | 41.2 | 38.6 | 36.8 |
| 80 | 100 | 100 | 52.7 | 46.3 | 40.4 | 35.5 | 32.4 | 31.3 |
| 60 | 100 | 57 | 48.3 | 41.9 | 35.6 | 30.6 | 27 | 25.7 |
| 40 | 100 | 56 | 46.9 | 39.2 | 32.1 | 26.7 | 22.4 | ex |
| 20 | 100 | 57 | 46.5 | 38.3 | 30.9 | 24.7 | ex | ex |
| 0 | 100 | 59 | 49.8 | 40.4 | 32.6 | 26.2 | ex | ex |
| -20 | 100 | 100 | 53.7 | 45.5 | 37 | 30.8 | 25.8 | 23.6 |
| -40 | 100 | 100 | 100 | 52 | 44 | 38 | 33.9 | 31.4 |
| -60 | 100 | 100 | 100 | 59.1 | 53 | 46 | 42.5 | 40.9 |
| -80 | 110 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

EX - Represents Non-Feasible Region



# APPENDIX B: PARTICIPATION IN AIAA YOUNG PROFESSIONAL ROCKET LAUNCH COMPETITION IN MAY 2009

## INTRODUCTION

The author participated in the AIAA Young Professional Rocket Launch Competition in May 2009 as a member of the NPS Team (see Figure 28 for the team photo). Team Peacock, one of the two teams from NPS, comprised eight student members and a faculty advisor (Prof Yakimenko). The competition required its participant to apply their knowledge to build a rocket, analyze and predict it performance and launch it at a test site, using common rocket kits, a choice of propulsion units and a payload provided by the organizers. The 8.5-feet long scaled model rocket was expected to reach height in excess of 7,000 at a speed of greater than Mach 1. The rocket has to be recovered after reaching its maximum height.

Figure 28. Team Members and Advisor with Their Rocket at the Launch Site

## RELEVANCE

The team’s participation in the competition is relevance to their course of study in NPS in general in that it provided an excellent opportunity for the students to apply their knowledge in aerodynamics, propulsion, missile design and manufacturing in analyzing a problems and physically implementing the solutions to achieve an actual task or a mission—in this case building the rocket and launching it. In the process, greater experience and knowledge was gained, not only in the academic areas, but also in terms of project management, teamwork and communication. This is a good example of the ‘hands-on’ approach towards learning and an interesting and impactful means of education. With respect to this project, this rocket is like a scaled model of an interceptor missile designed to intercept a ballistic missiles in the boost phase. The characteristics of the rocket resembles that of a hit-to-kill interceptor, vis high speed, ‘light weight’ (includes only the body and fins, actuators, guidance and control), small payload (perhaps a small KE warhead or no warhead at all) and ‘long range’ (high ratio of fuel mass to total mass). A 2-stage rocket would bear greater resemblance to a typical missile interceptor. The valuable experience gained provided the author greater insight and better understanding of the fundamental issues facing the design of interceptor missiles, particularly the aerodynamics, propulsion, stability and design trade-offs.

## COMPETITION

The judging criteria for the competition are the maximum height reached by the rocket and the accuracy of the predicted performance versus actual performance. There is a winner each for the highest height reached and the closest predicted-to-actual performance category. Besides the actual launch itself, a report submitted by each team on their analysis and work done is also assessed to determine the winner for the latter.

## GOALS

Team Peacock set itself two goals for this competition : (1) it aimed to be the winner for the highest altitude category and (2) to build the most stable rocket in flight and be able to recover the rocket successfully. This goal guided the team in its design and construction of the rocket.

## TASKS

The team took about 4 months to build the rocket and prepare for the launch, which was held in end May 2009. After the team was formed in Jan 09, it went straight to work by brainstorming what are the tasks to be completed, the timeline, resources required (including budget) and roles of individual members. Taking into account the goals and tasks identified, allocation of tasks and a schedule broad was drafted.

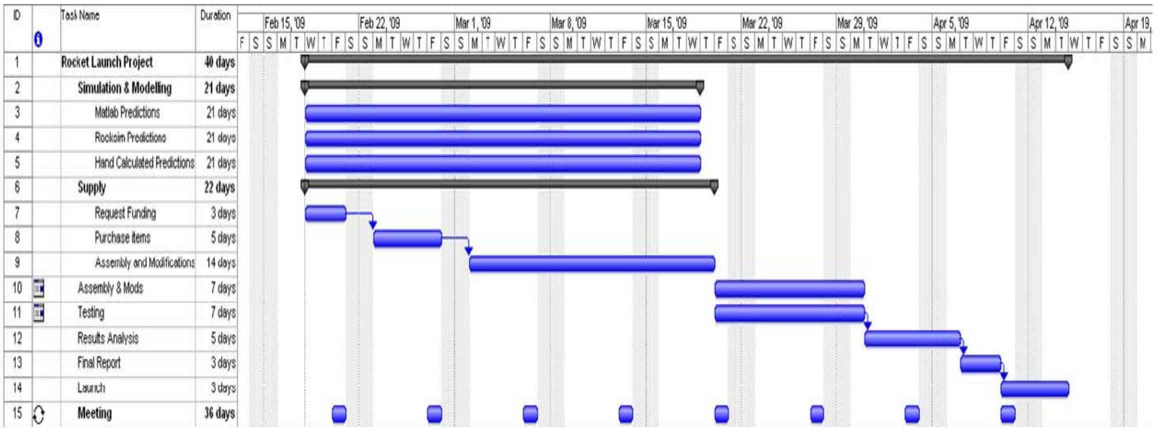
The broad tasks facing the team from the start to launch were as follows:

a. Budgeting. The team worked within a budget of $1000 to purchase all the required materials that is not provided by the organizer (only the rocket kit, propulsion unit and the common-use payload is provided). Table 8 provides a breakdown of the funds needed (for supplies).

| Item | Location | Cost($) | Qty | Total($) |
|---|---|---|---|---|
| 60" TAC-1 Parachute | Giant Leap Rocketry | 86 | 1 | 86 |
| TAC-1 Kevlar Bag | Giant Leap Rocketry | 25 | 1 | 25 |
| Tubular Kevlar Shock Cord (1/2" x 15') | Giant Leap Rocketry | 25 | 2 | 50 |
| 1/4" Eyelet Swivel | Giant Leap Rocketry | 3 | 4 | 12 |
| AeroTech Electronic Fwd Enclosure | Aerotech-Rocketry | 170 | 1 | 170 |
| Construction Materials | Perfectflite.com | 100 | 1 | 100 |
| K270W(engine) | Aerotech-Rocketry | 150 | 2 | 300 |
| Misc | | | | 217 |
| **TOTAL** | | | | 960 |

Table 8. Budget for Rocket Launch Competition

b. Scheduling. An initial schedule was drafted (see Figure 30) so that members have a good sense of the key milestones and timeline for the various preparatory activities. The schedule was updated as project progresses to reflect the status of the various tasks.



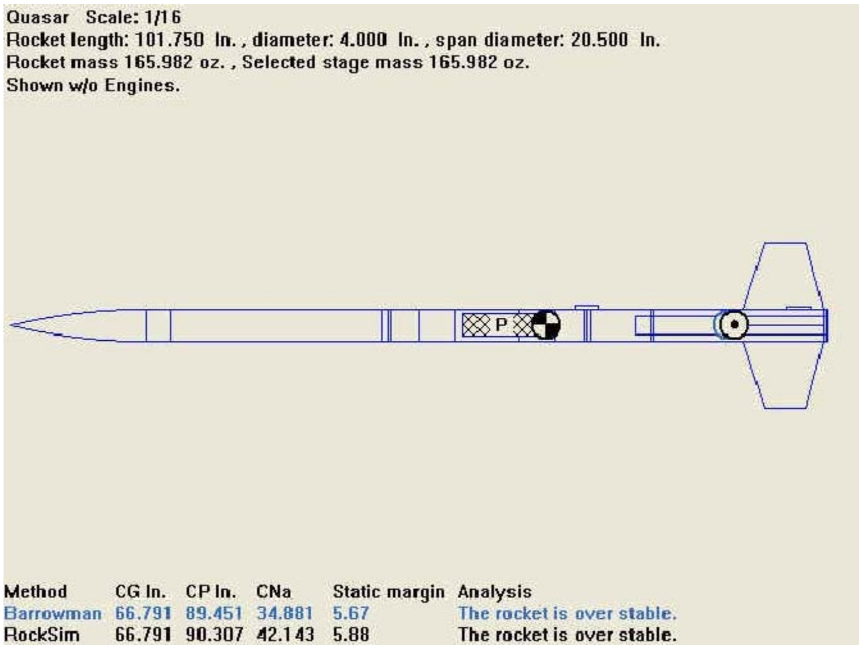
| ID | ℹ | Task Name | Duration |
|---|---|---|---|
| 1 | | **Rocket Launch Project** | **40 days** |
| 2 | | **Simulation & Modelling** | **21 days** |
| 3 | | Matlab Predictions | 21 days |
| 4 | | Rocksim Predictions | 21 days |
| 5 | | Hand Calculated Predictions | 21 days |
| 6 | | **Supply** | **22 days** |
| 7 | | Request Funding | 3 days |
| 8 | | Purchase items | 5 days |
| 9 | | Assembly and Modifications | 14 days |
| 10 | ▦ | Assembly & Mods | 7 days |
| 11 | ▦ | Testing | 7 days |
| 12 | | Results Analysis | 5 days |
| 13 | | Final Report | 3 days |
| 14 | | Launch | 3 days |
| 15 | ↻ | **Meeting** | **36 days** |

Feb 15, '09
Feb 22, '09
Mar 1, '09
Mar 8, '09
Mar 15, '09
Mar 22, '09
Mar 29, '09
Apr 5, '09
Apr 12, '09
Apr 19,
F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M

Figure 29. Timeline for the Rocket Launch Project

c. Resource Planning and Purchasing (Supply). Some of the team members were assigned the role to plan and decide the items, the specifications and quantities needed as well as to source and purchase the items (the actual purchasing is done by the administrative staff)

d. Structure Design. The rocket kit is provided by the organizers. It uses the Quasar 1/16 rocket, which comes with the nose cone, body sections, fins and also a parachute for recovery. The team used commercial software RockSim to predict the CG and CP calculation as well as to determine the structural dimensions for the rocket design. Figure 31 presents the CG and CP position and the static margin. Based on the calculation and prediction, the team decided to reduce the length of the rocket by 14 inches as well as halved the fin area to reduce the static margin to 2 and optimize the performance of the rocket so as to achieve the required stability and highest altitude.



Quasar Scale: 1/16
Rocket length: 101.750 In., diameter: 4.000 In., span diameter: 20.500 In.
Rocket mass 165.982 oz., Selected stage mass 165.982 oz.
Shown w/o Engines.

| Method | CG In. | CP In. | CNa | Static margin | Analysis |
|---|---|---|---|---|---|
| Barrowman | 66.791 | 89.451 | 34.881 | 5.67 | The rocket is over stable. |
| RockSim | 66.791 | 90.307 | 42.143 | 5.88 | The rocket is over stable. |

Figure 30. The Specifications and Dimensions of the Quasar Rocket Kit

e. Rocket Motor Selection. Two engines, both with about the same impulse, were provided for selection. The team can choose either the engine that has higher thrust and a shorter burn time (K800) or the other with lower thrust but longer burn time (K270). After conducting simulation study of the predicted flight profile, based on the rocket modified dimensions, using RockSim and a MATLAB program written by one of the team member, it was found that the K270 best meet our needs. The engine motor specifications are as follows:

| | |
|---|---|
| Motor | : AeroTech K270 |
| Diameter (mm) | : 54.0 |
| Length (cm) | : 57.9 |
| Prop. Weight (g) | : 1,188.0 |
| Total Weight (g) | : 2,100. |
| Avg. Thrust (N) | : 247.6 |
| Max. Thrust (N) | : 425.7 |
| Tot. Impulse (Ns) | : 2,154.9 |
| Burn Time (s) | : 8.7 |

The motor performance chart for the K270 are shown in Figure 32.

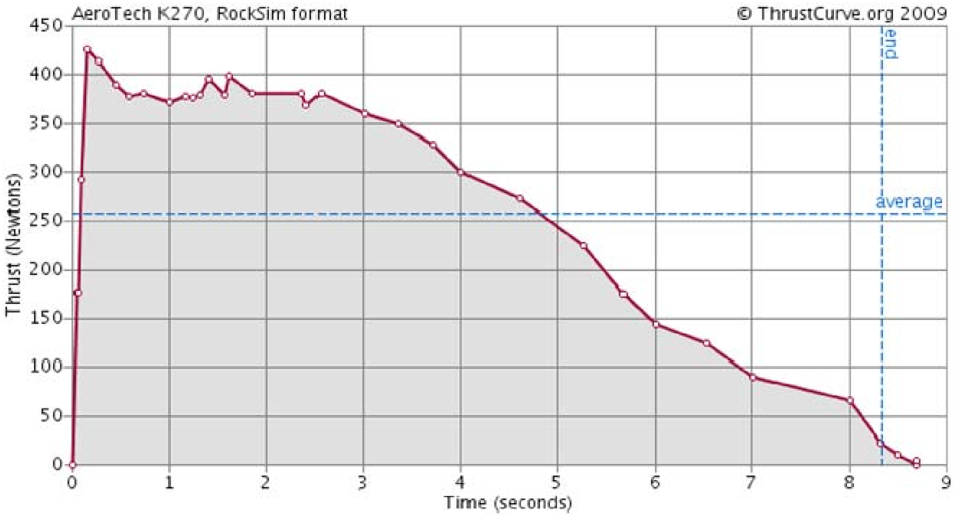


Figure 31. The Specifications and Dimensions of the Quasar Rocket Kit

f. Analysis (Simulation and Modeling). This is one of the key components of this rocket launch project. The team has to use various methods to analyze and predict the flight performance/profile of the rocket to be built. The team relied on several methods to analyze the flight performance of the rocket, including simple hand calculations for rough prediction, using RockSim, a commercial rocket software for amateur rocketeers and a student-developed MATLAB program (with Simulink model) written by one of our members. The analysis and simulation done helped the team to determine the structural modification needed, weight and balance, CG and CP position, stability, selection of engine and the maximum altitude prediction. Figure 33 shows the Simulink model. Comparison of results, derived from RockSim and own MATLAB model, can be made and it was found that they are within about 10% of each other. RockSim predicted a higher maximum height reached of 7,000 feet, while the Simulink model predicted 6,400 feet.

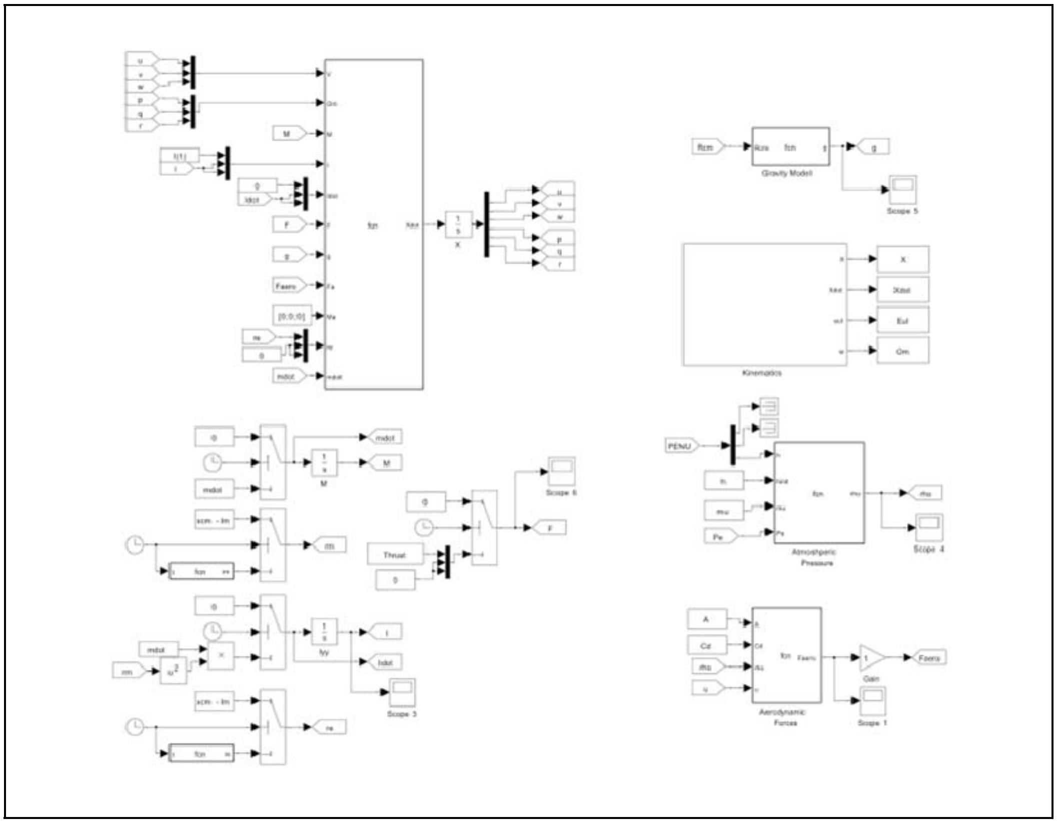


Figure 32. The Specifications and Dimensions of the Quasar Rocket Kit

g. Testing. Besides analysis, some of the rocket components have to be physically tested to have a higher assurance that it worked as designed and modifications made did not introduce problems to the design. The team tested the rocket engine at the propulsion laboratory to match the actual performance with manufacturer specifications, the Electronic Forward Closure (EFC) to test the trigger mechanism for the explosive charge to deploy the parachute and the actual parachute to select the one to be used for the rocket recovery system. During the first propulsion unit test, there was a burn-through of the engine casing due to a wrong part provided by the manufacturer. The problem was raised to the manufacturer and a replacement part was delivered. The discovery was an important one as it prevented the same occurrence from happening during the actual launch for our team as well as for other teams. The second firing was very successful and it verified the same propulsion characteristic given by the manufacturer. The EFC test verified that the trigger mechanism worked and would cause the separation of the rocket body after apogee to deploy the recovery chute. The parachute testing helped the team to decide using a single chute system for recovery instead of a dual-chute system.  Figure 34 shows pictures of testing being conducted.

Figure 33. Team Members and Advisor with Their Rocket at the Launch Site

h. Construction. The actual construction of the rocket took about three weeks, after all the parts and materials were delivered. For a start, a rocket stand or jig was build to facilitate the assembling work. For this rocket, the body was made up of several cylindrical sections of thick cardboard material and the nose cone is made of plastic. The construction process begins with strengthening the body with layers of epoxy. The internal compartments and fins were joined to the body using fibre-glass cloth with epoxy and left to cure. The various structures were attached to the rocket body or inserted inside the rocket in sections. Fixing the fins to the tail section required more attention to ensure they are symmetrically attached. Much effort was put in to ensure the rigidity as the fins as they would be subjected to relatively large forces in flight. The rocket engine casing, EFC and parachute were then put inside the rocket body. Once all the components were assembled in their sections, the three sections were then joined together using internal connectors and nose cone was attached to the body. The last step is painting the rocket to give it a distinctive color for aesthetic reasons as well as to provide a smooth finishing to reduce skin drag. The payload would be placed inside the rocket only on launch day. It consisted of an altimeter that would send altitude data continuously to the ground receiver for recording purposes. Figure 37 shows some pictures of the rocket being built at the laboratory.

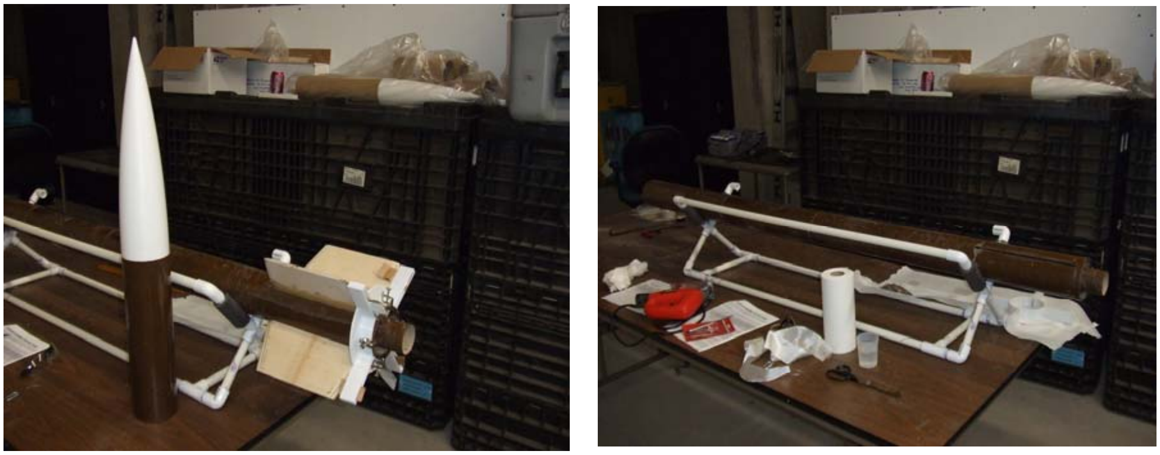


Figure 34. Team Members and Advisor with Their Rocket at the Launch Site

i. Launch. The launch was the climax of the five-month effort and the team was as prepared as it could be. It was held on 26 May 09, Saturday at Koehn Lake Launch Site near to Mojave. On that day, there were 9 teams present with their rockets. Team Peacock was the first team to launch. After the rocket propellant was inserted into the casing, it was mounted onto the launch rail for the ignition. The igniter was fired by the firer, who was positioned at the firing ‘bunker’. All other observers were herded to the observation deck to watch the firing. At the countdown of 10, the rocket was launched. It lifted-off vertically with a sharp ‘bang’ and within seconds, the rocket motor accelerated the rocket to more than 3,000 ft and out of visual sight, leaving a trail of smoke behind it. Moments later, the parachute was deployed and the rocket descended to the ground in two parts, as expected. It landed ‘safely’ at the desert ground close to the launch site and was retrieved by the recovery team. Figure 36 shows the launch of the rocket.

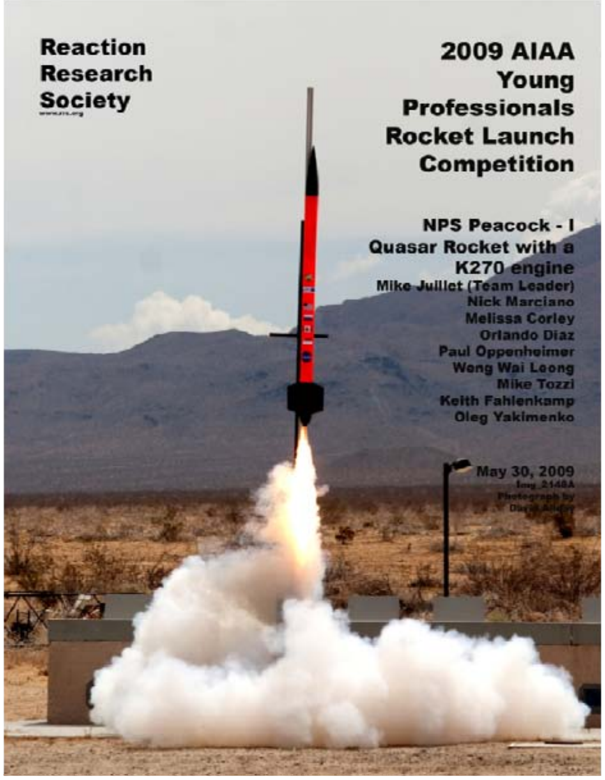


Figure 35. Team Members and Advisor with Their Rocket at the Launch Site

## RESULTS

The results made all the hard work pay off. Team Peacock emerged the winner of the closest altitude prediction category. The recorded maximum altitude reached during the actual launch was 5,700 feet. This was about 9% deviation from the predicted height based on the team’s analysis. Figure 37 presents the results in terms of the altitude versus time plot that was recorded by the organizer.

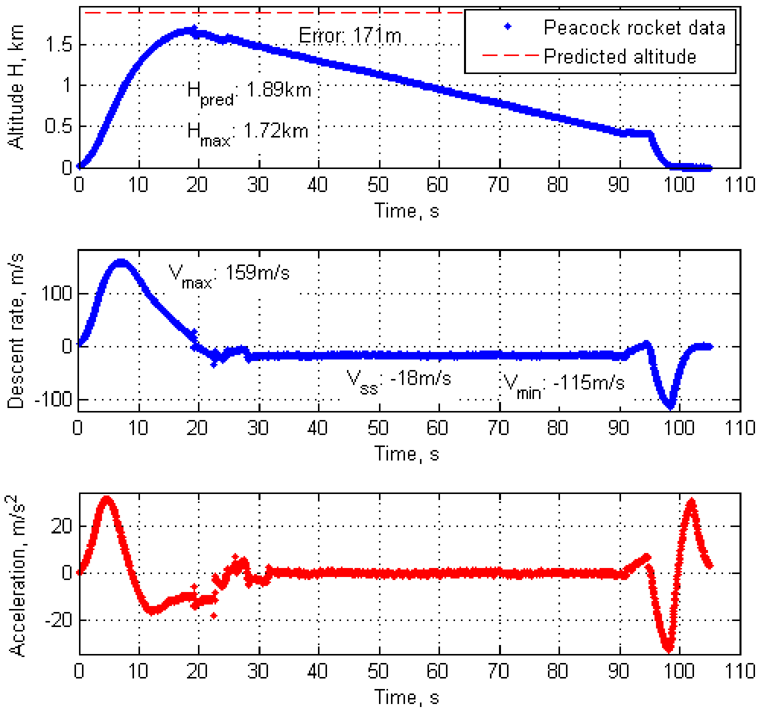


Figure 36. Official Altitude Plot for Team Peacock

## LESSON LEARNED

The end result is not as important as the process of getting it. Though the team did not expect to win and entered the competition with an altitude to experience and understand rocket design and rocketry better, the results was nonetheless a good one for first-timers like us. There were many lessons learnt from the success and failure of some teams. By analyzing and discussing the causes of failure with the other teams, it provided much insight into the considerations that went into the design and some of the cardinal errors that were made. Using the example of a rocket that ‘exploded’ mid-air, their over-ambitious fin design, which reduced too much of the control surface for normal flight and stability, it provided a good lesson for our team as well in our future design. In future participation, more prediction tools and program should be made available for team members to analysis the aerodynamics and flight performance. The project may be part of the NPS missile design course, culminating in the actual launch of the rocket; all these within 2 quarters, It can help to reinforce some of the fundamental principles of aerodynamics and rocket design. In summary, this was a worthwhile project that combined learning, application of knowledge and great fun into one, and certainly helped in providing a better grasp of fundamentals in missile design.

# APPENDIX C: INDUCED DRAG PROGRAMME DEVELOPED

## 1. ZLDragC_mod.m

```
function [CD] = ZLDragC_mod(M,load_f,mass,rho,vel,Sref)
% Written by LTC Weng Wai Leong, Naval Postgraduate School, Dec 2009

% This function interpolates the known drag polars for various Mach
number to determine drag coefficient. Input variables are Mach number,
% load factor, mass, density, velocity, referenece area the interceptor
or ballistic missile (BM) to be utilized
% in the BMFlight3.m, SMTrajectory and SMFlight3.m programs (written by
LT Lukacs) for the calculation of forces acting on the
interceptor/Ballistic Missile.

% The CL and Cd data are input in Excel files. Data obtained from Prof
Yakimenko

% Variable List
% M = mach number of interceptor/BM
% load_f= load factor in the normal plane
% mass = mass of interceptor/BM
% rho = density of air
% vel = velocity of interceptor/BM
% Sref = reference area of interceptor/BM

global CL_data Cd_data

Cd_curve = interp1(Cd_data(:,1),Cd_data(:,2:11),M,'nearest','extrap');
%interpolate the Cd data in the array for the new Mach number

CL_curve = interp1(CL_data(:,1),CL_data(:,2:11),M,'nearest','extrap');
%interpolate the CL data in the array for the new Mach number

p = polyfit(CL_curve,Cd_curve,2);    % generate a new drag polar (CL_Cd
curve) for new Mach number using 2nd order polynominal function

k=p(1);

CL0=0;

CD0=p(3)-k*CL0^2;

CL=(2*load_f*mass*9.81)/(rho*vel^2*Sref);

if abs(CL)>5, CL=5*sign(CL); end

CD = CD0 + k*(CL-CL0)^2;
return
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX D: MODELING PROGRAM CODE

## A. 3DOF INTERCEPTOR

### 1. SMFlight3.m

```
%% This is the Main M-File for the simulation. Run this file with all
the required function files in the same current directory.

% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006
% Corrected by O.Yakimenko, August 2007, November 2009

% This script develops and tracks the flight path of the interceptor
% missile. For the first ten seconds it integrates a series of
% acceleration commands to simulate a vertical launch and tip over.
Upon
% activation of the guidance law, it sends the known values to the
guidance
% law and receives back the future time history of the optimal flight
path.
% This script then implements that optimal path. It updates the final
% conditions and recalculates the optimal flight path at an interval of
10
% seconds. The script calls BRFlight3, SMParams3.m, SMDrag.m,STatmos.m,
% and SMGuidance.m

%% List of variables
% Acc_SM     = index-based vector of acceleration values
% AllSM      = index based vector of all interceptor values
% alt        = altitude
% CD         = drag coefficient
% count        = counting variable to determine guidance law update
interval
% dist       = cumulative distance travelled
% Drag       = total drag force
% Forces_SM  = index-based vector of force values
% g              = gravitational force, based on WGS-84 value of
gravitational
%               attraction and altitude
% m_i        = interceptor mass
% Model_SM   = index-based vector of internal values
% MV          = interceptor speed in Mach (relative to local speed of
sound)
% N1,N2       = variables used to ensure optimal vectors are the same
length
% num_SM     = number of interations conducted (used for plotting)
% nx,ny,nz    = axial acceleration command, body frame x, y and z,
respectively
% nx(y,z)_Op = index based vector of the optimal flight path values
% path       = returned time history of the optimal path
% Pos_SM     = index-based vector of position values
% press      = local atmospheric pressure
% psi         = heading angle
% psidot      = rate of change of heading angle
% psidot_Op   = index based vector of the optimal flight path values
% px,py,pz    = x, y and z components of position
% px(y,z)_Op = index based vector of the optimal flight path values
% py_Op       = index based vector of the optimal flight path values
% pz_Op       = index based vector of the optimal flight path values
% q           = counting variable (used in another program)
% Re          = WGS-84 Earth's radius
% ro          = local atmospheric density
% Sref        = planar reference area (for drag calculations)
% state       = the state of the interceptor
% t           = global time
% target      = the state of the rocket
% temp        = local atmospheric temperature
% tgo         = time to go to intercept
% th          = flight path angle
% th_Op       = index based vector of the optimal flight path values
% thdot       = rate of change of flight path angle
% thdot_Op    = index based vector of the optimal flight path values
% Thrust      = thrust generated by interceptor motor
% time_Op     = index based vector of the optimal flight path values
% time_SM     = index-based vector of time values
% update      = number of updates to the guidance law conducted
% V           = velocity of the interceptor
% V_Op        = index based vector of the optimal flight path values
% Vdot        = rate of change of the velocity
% Vdot_Op     = index based vector of the optimal flight path values
% Vel_SM      = index-based vector of velocity values
% w             = number of 0.5s steps between the lunches of traget and
interceptor

close all, clear all, clc

global Re alphag path              % shared by SMFlight3, SMGuidance &
SMTrajectory
global q states                    % shared by SMFlight3 & SMGuidance
global  Pos_BR  Pos_SM  Npol  Npt  kpolar  weight90  %  ...  by  SMFlight3  &
SMTrajectory

%% Computing the flight path of a Ballistic Missile in a gravity turn
BRFlight3

%% Initializing variables for an Interceptor
t=0; dt=0.5; q=0; w=120; % 60 sec detection time
Nupd=30; count=Nupd; update=0;
Npt=100;              % the number of points the optimal trajectory is
computed at
Npol=8;                    %  number  of  coefficients  in  approximating
polynomials
kpolar=0;

prompt = {'Northing wrt to BM launch point, km',...
           'Westing wrt to BM launch point, km',...
           'Weighting coefficient for impact angle'};
 dlg_title = 'Enter Interceptor launch coordinates';
 num_lines = 1;
 def = {'80','60','1000'};
 answer = inputdlg(prompt,dlg_title,num_lines,def,'on');
 px      =str2num(answer{1})*1000;
 py      =str2num(answer{2})*1000;
 weight90=str2num(answer{3});
% px=-100000/20;
% py=100000/2;
% weight90=100;
pz=Re;
px_old=px; py_old=py; pz_old=pz;
dist=0;

psi=atan2(Pos_BR(120,2)-py,Pos_BR(120,1)-px); psi_old=psi;
th=90*pi/180; th_old=th;

V=1; V_old=V;

%% Computing Interceptor flight path
for i=1:20%1000
    t=t+dt;
%% Boost phase (vertical launch), t<10s
if t<10
    % Speed, Mach number
    alt=norm([px;py;pz])-Re;
    [ro,press,temp]=STatmos(alt);
    MV = V/sqrt(1.402*287.053*temp);

    % Forces
    g=3.986004418e14/norm([px;py;pz])^2;
    [m_i,Sref] = SMParams3(t);
    CDtable = SMDrag(MV);
        if t<=6,    Thrust = 206000; psidot=0;        thdot=0;
        else          Thrust = 95300;   psidot=0;          thdot=-0.075;
end
    ny = V/g*cos(th)*psidot;
    nz = V/g*thdot+cos(th);
    nn = sqrt(ny^2+nz^2);
        CL=(2*nn*m_i*g)/(ro*V^2*Sref);
        CD = CDtable(1) + kpolar*CL^2;
    Drag = ro*V^2*CD*Sref/2;
    nx=(Thrust-Drag)/m_i/g;

    alphag=180/pi*nn*m_i*g/(ro*V^2*Sref/2)/13;

    % Kinematics
    Vdot=g*(nx-sin(th));
    psidot=ny*g/V/cos(th);
    thdot=g*(nz-cos(th))/V;

    % Collecting variables
    time_SM(i,1)=t;
    Forces_SM(i,1)=nx;     Forces_SM(i,2)=ny;     Forces_SM(i,3)=nz;
    Model_SM(i,1)=V;       Model_SM(i,2)=Vdot;
    Model_SM(i,3)=th;      Model_SM(i,4)=thdot;
    Model_SM(i,5)=psi;     Model_SM(i,6)=psidot;
    Pos_SM(i,1)=px;        Pos_SM(i,2)=py;        Pos_SM(i,3)=pz;
    Pos_SM(i,4)=dist;
    Vel_SM(i,1)=V*cos(th)*cos(psi);
    Vel_SM(i,2)=V*cos(th)*sin(psi);
    Vel_SM(i,3)=V*sin(th);
    Acc_SM(i,1)=Vdot*cos(th)*cos(psi)-V*cos(th)*sin(psi)*psidot...
        -V*sin(th)*cos(psi)*thdot;
    Acc_SM(i,2)=Vdot*cos(th)*sin(psi)+V*cos(th)*cos(psi)*psidot...
        +V*sin(th)*sin(psi)*thdot;
    Acc_SM(i,3)=Vdot*sin(th)+V*cos(th)*thdot;

    % Euler integration
    V=V_old+Vdot*dt;
    psi=psi_old+psidot*dt;
    th=th_old+thdot*dt;
    px=px_old+V*cos(th)*cos(psi)*dt;
    py=py_old+V*cos(th)*sin(psi)*dt;
    pz=pz_old+V*sin(th)*dt;

    dist=(dist+abs(norm([px-px_old;py-py_old;pz-pz_old])));

    V_old=V;      psi_old=psi;  th_old=th;
    px_old=px;    py_old=py;    pz_old=pz;

else
%% Optimal guidance, t>10s
    if count==Nupd & update==0      % Recomputing the trajectory every
Nupd cycles
        update=update+1;
        fprintf('Starting Interceptor''s Guidance Update
#%2.0f\n',update)
        state=[px;py;pz;V;th;psi;Vdot;thdot;psidot];
        target=[Pos_BR(i+w,1);Pos_BR(i+w,2);Pos_BR(i+w,3);
                Vel_BR(i+w,1);Vel_BR(i+w,2);Vel_BR(i+w,3);
                Acc_BR(i+w,1);Acc_BR(i+w,2);Acc_BR(i+w,3)];
        path=SMGuidance(t,state,target,i);         % Calling SMGuidance
function
            if update==1;
                N1=length(path(:,1)); N2=N1;
            else
                N2=length(path(:,1));
            end
        % Identifying variables
        time_Op(:,update)=[path(:,1);zeros(N1-N2,1)];
        px_Op(:,update)=[path(:,2);zeros(N1-N2,1)];
        py_Op(:,update)=[path(:,3);zeros(N1-N2,1)];
        pz_Op(:,update)=[path(:,4);zeros(N1-N2,1)];
        V_Op(:,update)=[path(:,5);zeros(N1-N2,1)];
        th_Op(:,update)=[path(:,6);zeros(N1-N2,1)];
        psi_Op(:,update)=[path(:,7);zeros(N1-N2,1)];
%         Vdot_Op(:,update)=[path(:,8);zeros(N1-N2,1)];
%         thdot_Op(:,update)=[path(:,9);zeros(N1-N2,1)];
%         psidot_Op(:,update)=[path(:,10);zeros(N1-N2,1)];
        nx_Op(:,update)=[path(:,8);zeros(N1-N2,1)];
        ny_Op(:,update)=[path(:,9);zeros(N1-N2,1)];
        nz_Op(:,update)=[path(:,10);zeros(N1-N2,1)];
        count=1;
    end
            if     count==101            % added by OY 2009-10-08
                   count=count-1;
            elseif count==0
                   count=1;
            end

    t=time_Op(count,update);
    nx=nx_Op(count,update);
    ny=ny_Op(count,update);
    nz=nz_Op(count,update);
    V=V_Op(count,update);
%     Vdot=Vdot_Op(count,update);
    th=th_Op(count,update);
%     thdot=thdot_Op(count,update);
    psi=psi_Op(count,update);
%     psidot=psidot_Op(count,update);
    px=px_Op(count,update);
    py=py_Op(count,update);
    pz=pz_Op(count,update);

    % Collecting variables
    time_SM(i,1)=t;
    Forces_SM(i,1)=nx;     Forces_SM(i,2)=ny;     Forces_SM(i,3)=nz;
    Model_SM(i,1)=V;       Model_SM(i,2)=Vdot;
    Model_SM(i,3)=th;      Model_SM(i,4)=thdot;
    Model_SM(i,5)=psi;     Model_SM(i,6)=psidot;
    Pos_SM(i,1)=px;        Pos_SM(i,2)=py;        Pos_SM(i,3)=pz;
    Pos_SM(i,4)=dist;
    Vel_SM(i,1)=V*cos(th)*cos(psi);
    Vel_SM(i,2)=V*cos(th)*sin(psi);
    Vel_SM(i,3)=V*sin(th);
    Acc_SM(i,1)=Vdot*cos(th)*cos(psi)-V*cos(th)*sin(psi)*psidot...
        -V*sin(th)*cos(psi)*thdot;
    Acc_SM(i,2)=Vdot*cos(th)*sin(psi)+V*cos(th)*cos(psi)*psidot...
        +V*sin(th)*sin(psi)*thdot;
    Acc_SM(i,3)=Vdot*sin(th)+V*cos(th)*thdot;
    Update(i,1)=update;

    % Time step
    count=count+1;
    dist=(dist+abs(norm([px-px_old;py-py_old;pz-pz_old])));
    V_old=V;
    psi_old=psi;
    th_old=th;
    px_old=px;
    py_old=py;
    pz_old=pz;
end                     % the end of the "if" loop
end                      % the end of the "for" loop

AllSM= [time_SM Forces_SM Model_SM Pos_SM Vel_SM Acc_SM]; % update];
```

### 2. SMParam3.m

```
function [SM_mass,Sref]=SMParams3(t)
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006

% This function calculates the J Matrix and Mass of the intercepter,
% assuming a cruciform rocket in two stages to intercept.  This
% function also returns the reference (base) diameter of the missile.

%% Notes:
% The first stage last 6 seconds, the second stage lasts an additional
% 10 seconds.  Stage 1 (booster) seperates upon completion.  Stage 2
% does not separate after completion

%% List of variables
% dia           = reference diameter, base diameter
% l             = length, varies by component
% p_SM_st1_fuel = density of stage 1 rocket fuel
% p_SM_st2_fuel = density of stage 2 rocket fuel
% p_SMstr       = density of structural material
% r             = radius, varies by component
% ro            = outer radius, varies by component
% ri            = inner radius, varies by component
% SM_mass       = total rocket mass
% SM_nose       = total mass of nosecone section
% SM_st1_fcr    = consumption rate of stage 1 fuel
% SM_st1_fuel   = remaining stage 1 fuel based on time and
%                 consumption rate
% SM_st1_str    = total mass of stage 1 structural material
% SM_st1_tfm    = total mass of stage 1 fuel
% SM_st2_fcr    = consumption rate of stage 2 fuel
% SM_st2_fuel   = remaining stage 2 fuel based on time and
%                 consumption rate
% SM_st2_str    = total mass of stage 2 structural material
% SM_st2_tfm    = total mass of stage 2 fuel
% t             = time
% th            = structural thickness
% V_bo dy        = volume of body structural material
% V_nose_str    = volume of nosecone structural material
% V_nose_str0   = volume of nosecone structural material,
%                 intermediate value
% V_nose_str1   = volume of nosecone structural material,
%                 intermediate value
% V_st1_fuel    = volume of stage 1 fuel
% V_st1_str     = volume of stage 1 structural material
% V_st2_fuel    = volume of stage 2 fuel
% V_st2_str     = volume of stage 2 structural material

%% Structural components
    p_SMstr = 4225;
    th = .0208;

    % Mass of nose cone
    l = .8255;
    r = 0.34/2;
    V_nose_str0 = pi*(l*((r^2+l^2)/(2*r))^2-l^3/3-(((r^2+l^2)/...
          (2*r))-r)*((r^2+l^2)/(2*r))^2*asin(l/((r^2+l^2)/(2*r))));
    l = .8255-th;
    r = 0.34/2-th;
    V_nose_str1 = pi*(l*((r^2+l^2)/(2*r))^2-l^3/3-(((r^2+l^2)/...
          (2*r))-r)*((r^2+l^2)/(2*r))^2*asin(l/((r^2+l^2)/(2*r))));
    V_nose_str = V_nose_str0-V_nose_str1+pi*r^2*th;
    SM_nose = 1.3*V_nose_str*p_SMstr;

    % Mass of Body/Warhead Section
    l = .849;
    ro = 0.34/2;
    ri = 0.34/2-th;
    V_body = l*pi*(ro^2-ri^2);
    SM_body = V_body*p_SMstr+115;

    % Mass of Stage 1 (Mk72 Booster)
    l = 1.72;
    ro = 0.53/2;
    ri = 0.53/2-th;
    V_st1_str = l*pi*(ro^2-ri^2)+2*pi*ro^2*th;
    SM_st1_str = V_st1_str*p_SMstr;

    % Mass of Stage 2 (Mk104 Engine)
    l = 2.88-2*th;
    ro = 0.34/2;
    ri = 0.34/2-th;
    V_st2_str = l*pi*(ro^2-ri^2)+2*pi*ro^2*th;
    SM_st2_str = V_st2_str*p_SMstr;

%% Fuel Components
    % Stage 1 Solid Fuel is HTPB/AP/Al
    l = 1.72;
    ri = 0.53/2-th;
    V_st1_fuel = 0.80*l*pi*ri^2;
    p_SM_st1_fuel = 1860;
    SM_st1_tfm = 468;
    SM_st1_fcr = 468/6;
    % Stage 2 Solid Fuel is TP-H1205/6
    l = 2.88;
    ri = 0.34/2-th;
    V_st2_fuel = 0.60*l*pi*ri^2;
    SM_st2_tfm = 360;
    p_SM_st2_fuel = SM_st2_tfm/V_st2_fuel;
    SM_st2_fcr = 360/15;

if t<6
    %% Stage 1 - Stage 1 Fuel is consumed, Stage 2 Fuel is not used.
    SM_st1_fuel = SM_st1_tfm - SM_st1_fcr * t;
    SM_mass = SM_nose+SM_body+SM_st2_str+SM_st2_tfm+SM_st1_str+...
          SM_st1_fuel;
    dia = 0.53;

elseif t<21
    %% Stage 2 - Stage 1 has seperated, Stage 2 Fuel is consumed.
    SM_st2_fuel = SM_st2_tfm - SM_st2_fcr * (t-6);
    SM_mass = SM_nose+SM_body+SM_st2_str+SM_st2_fuel;
    dia = 0.34;

else
    %% Stage 3 - The unpowered nosecone and Stage 2 remains.
    SM_mass = SM_nose+SM_body+SM_st2_str;
    dia = 0.34;

end

Sref=pi*dia^2/4;

return
```

### 3. SMDrag.m

```
Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006
and renamed by Prof Oleg Yakimenko in November 2009

% This function interpolates to determine drag coefficient based on the
% Mach number and the boost or glide phase of the rocket to be utilized
% in the BMFlight.m and SMFlight.m programs for the calculation of
% forces acting on the rocket/missile.

% The tables used were point-plotted from Prof Hutchins' ME4703
% "Missile Flight Analysis" Class Notes

%% Setting a List of Variables

% BDrag = boost phase drag interpolation table
% GDrag = glide phase drag interpolation table
% M = mach number
% Mach = mach number interpolation table

%% Tables:
Mach =  [0      0.90   1.1    1.2    1.5    2.0    2.5    3.0...
         3.5    5.0    6.0];
BDrag = [0.1444 0.1444 0.2778 0.2778 0.2308 0.1778 0.1481 0.1296...
```

```
         0.1185 0.1000 0.0950];
GDrag = [0.2461 0.2461 0.4615 0.4615 0.3615 0.2846 0.2500 0.2192...
         0.2000 0.1500 0.1300];
%plot(Mach,BDrag,'o--',Mach,GDrag,'+-.')

%% Calculation of Drag Coefficient Values:
if M > 6
   M = 6;
end
BDrag = interp1(Mach,BDrag,M,'cubic');
GDrag = interp1(Mach,GDrag,M,'cubic');
Drag=[BDrag;GDrag];
return
```

## B. 3DOF TARGET

### 1. BRFlight3.m

```
%% Complementary M-File
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006

%  This  script  integrates  the  position,  velocity,  and  acceleration
values
% at each time step to determine the flight path of a ballistic missile
% in a gravity turn.  The script calls BRParams3.m, ZLDragC.m, and
% STatmos.m

%% Setting a List of Variables

% acc       = total acceleration
% alt       = altitude
% ax        = x component of acceleration
% ay        = y component of acceleration
% az        = z component of acceleration
% CD        = drag coefficient
% Drag      = total drag force
% dt        = time step interval
%  g              =  gravitational  force,  based  on  WGS-84  value  of
gravitational
%             attraction and altitude
% gm        = initial launch angle
% i         = interval count
% m_r       = rocket mass
% Mspd      = rocket speed in Mach (relative to local speed of sound)
% num_BR    = number of interations conducted (used for plotting)
% nx        = axial force
% press     = local atmospheric pressure
% px        = x component of position
% py        = y component of position
% pz        = z component of position
% Re         = WGS-84 value for Earth's radius
% ro         = local atmospheric density
% spd        = rocket speed in m/s
% Sref       = planar reference area (for drag calculations)
% t          = time
% temp       = local atmospheric temperature
% Thrust     = thrust generated by motor
% vx         = x component of velocity
% vy         = y component of velocity
% vz         = z component of velocity
% Acc_BR     = index-based vector of acceleration values
% Forces_BR = index-based vector of force values
% Pos_BR     = index-based vector of position values
% time_BR    = index-based vector of time values
% Vel_BR     = index-based vector of velocity values
% All_BR     = index based vector of all rocket values

%% Initializing Variables
dt=0.5; i=0;
Re=6.378137e6;

%% Setting Initial Conditions
px=0;
py=0;
pz=Re;
gm=75*pi/180;
vx=cos(gm);
vy=0;
vz=sin(gm);

%% Computing Ballistic Flight Path
for t=0:dt:200%22%19.5
    i=i+1;

    % Speed, Mach Number
    spd=norm([vx;vy;vz]);
    alt=norm([px;py;pz])-Re;
    if alt<86000
        [ro,press,temp]=STatmos(alt);       % Calling STatmos function
    else
        [ro,press,temp]=STatmos(86000);     % Calling STatmos function
    end
    Mspd=spd/sqrt(1.402*287*temp);

    % Forces
    g=3.986004418e14/norm([px;py;pz])^2;
    [m_r,Sref]=BRParams3(t);                        %  Calling  BRParams3
function
    CD=BRDrag(Mspd);                        % Calling BRMrag function

    if t<125
        Thrust=105000*9.81;
        CD=CD(1);
    elseif t<240
```

```
        Thrust=29950*9.81;
        CD=CD(1);
    else
        Thrust=0;
        CD=CD(2);
    end
    Drag=ro*spd^2*CD*Sref/2;
    nx=(Thrust-Drag)/m_r/g;

    % Accelerations
    g=3.986004418e14*pz/norm([px;py;pz])^3;
    ax=g*nx*cos(gm);
    ay=0;
    az=g*nx*sin(gm)-g;
    acc=norm([ax;ay;az]);

    % Collect Variables
    time_BR(i,1)=t;
    Pos_BR(i,1)=px;
    Pos_BR(i,2)=py;
    Pos_BR(i,3)=pz;
    Pos_BR(i,4)=norm([px;py;pz]);
    Vel_BR(i,1)=vx;
    Vel_BR(i,2)=vy;
    Vel_BR(i,3)=vz;
    Vel_BR(i,4)=spd;
    Vel_BR(i,5)=Mspd;
    Acc_BR(i,1)=ax;
    Acc_BR(i,2)=ay;
    Acc_BR(i,3)=az;
    Acc_BR(i,4)=acc;
    Acc_BR(i,5)=nx;
    Forces_BR(i,1)=Thrust;
    Forces_BR(i,2)=m_r;
    Forces_BR(i,3)=Drag;

    % Time Step
    px=px+dt*vx;
    py=py+dt*vy;
    pz=pz+dt*vz;
    vx=vx+dt*ax;
    vy=vy+dt*ay;
    vz=vz+dt*az;
    num_BR=length(time_BR);
end
%plot(time_BR(:,1),(Pos_BR(:,3)-Re)/1000)
AllBR= [time_BR Forces_BR Pos_BR Vel_BR Acc_BR];
clear CD Drag Mspd Sref Thrust acc alt ax ay az dt
clear g gm h i m_r nx press px py pz ro spd t temp vx vy vz
```

### 2. BRParams3.m

```
function [BR_mass,Sref,length]=BRParams(t)
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006

% This function calculates the mass of the rocket, assuming a cruciform
% rocket in two stages plus an unpowered nosecone stage.  This function
% also returns the reference (base) diameter of the missile.

%% Notes:
%  The  first  stage  last  125  seconds,  the  second  stage  lasts  an
additional
% 110 seconds.  The stages seperate upon completion.

%% Setting a List of Variables
% BR_mass   = total rocket mass
% BR_nose   = total mass of nosecone section
% BR_st1_bt = burntime for stage 1
% BR_st1_fcr    = consumption rate of stage 1 fuel
% BR_st1_fuel    = remaining stage 1 fuel based on time and consumption
rate
% BR_st1_str    = total mass of stage 1 structural material
% BR_st1_tfm    = total mass of stage 1 fuel
% BR_st2_bt = burntime for stage 2
% BR_st2_fcr    = consumption rate of stage 2 fuel
% BR_st2_fuel    = remaining stage 2 fuel based on time and consumption
rate
% BR_st2_str    = total mass of stage 2 structural material
% BR_st2_tfm    = total mass of stage 2 fuel
% dia           = reference diameter, base diameter
% t     = time

%% Setting Structural Components
    BR_nose = 250;
    BR_st2_str = 2288;
    BR_st1_str = 9000;

%% Setting Fuel Components
    BR_st1_tfm = 50970;
    BR_st1_bt = 125;
    BR_st1_fcr = BR_st1_tfm/BR_st1_bt;
    BR_st2_tfm = 12912;
    BR_st2_bt = 110;
    BR_st2_fcr = BR_st2_tfm/BR_st2_bt;

if t<125
%% Stage 1 - Stage 1 Fuel is consumed, Stage 2 Fuel is not used
    BR_st1_fuel = BR_st1_tfm-BR_st1_fcr*t;
    BR_mass = BR_nose+BR_st1_str+BR_st1_fuel+BR_st2_str+BR_st2_tfm;
    dia = 2.2;
    length = 2+14+16;

elseif t<240;
%% Stage 2 - Stage 1 has seperated, Stage 2 Fuel is consumed
```

```
    BR_st2_fuel = BR_st2_tfm-BR_st2_fcr*(t-125);
    BR_mass = BR_nose+BR_st2_str+BR_st2_fuel;
    dia = 1.3;
    length = 2+14;

else
%% Stage 3 - Stage 2 has seperated, only the unpowered nosecone remains
    BR_mass = BR_nose;
    dia = 1.3;
    length = 2;

end
Sref=pi*dia^2/4;
return
```

### 3. BRDrag.m

```
function Drag = BRDrag(M)
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006

% Renamed by Oleg Yakimenko, November 2009 to be used as separate drag
file
% for interceptor and ballistic missile instead of the common ZLDragC.m
file
% used previously

% This function interpolates to determine drag coefficient based on the
% Mach number and the boost or glide phase of the rocket to be utilized
% in the BMFlight.m and SMFlight.m programs for the calculation of
% forces acting on the rocket/missile.

% The tables used were point-plotted from Prof Hutchins' ME4703
% "Missile Flight Analysis" Class Notes

%% List of variables

% BDrag = boost phase drag interpolation table
% GDrag = glide phase drag interpolation table
% M = mach number
% Mach = mach number interpolation table

%% Tables:
Mach =  [0      0.90   1.1    1.2    1.5    2.0    2.5    3.0...
         3.5    5.0    6.0];
BDrag = [0.1444 0.1444 0.2778 0.2778 0.2308 0.1778 0.1481 0.1296...
         0.1185 0.1000 0.0950];
GDrag = [0.2461 0.2461 0.4615 0.4615 0.3615 0.2846 0.2500 0.2192...
         0.2000 0.1500 0.1300];
```

```
%plot(Mach,BDrag,'o--',Mach,GDrag,'+-.')

%% Calculation of drag coefficient for the boost and glide phases:
if M > 6
   M = 6;
end
BDrag = interp1(Mach,BDrag,M,'cubic');
GDrag = interp1(Mach,GDrag,M,'cubic');
Drag=[BDrag;GDrag];
return
```

## C. GUIDANCE ALGORITHMS

### 1. SMGuidance.m

```
function path=SMGuidance(time,state,target,i)
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006
% Corrected by O.Yakimenko, August 2007, October 2009

% This function takes in the state of the interceptor and target and
% generates an initial guess at the final conditions (position,
% orientation angles, range, and time to intercept) through a first-
order
% trajectory assumption and iterative process.  It then calls the
% fminsearch function using those initial guesses.  Finally, it plots
the
% returned optimal flight path and associated variables.

%% List of variables
% best      = vector of the variables in the optimal path returned from
%             the fminsearch function
% BC        = boundary conditions
% cost      = cost function value returned from SMGuidanceCost function
%  costs       =  array  of  the  value  of  the  cost  variables  at  each
iteration
% free      = variables that fminsearch can modify, specifically
%             [tau;thf;psif]
% init      = vector of initial estimates
% J         = vector of cost function variable values
% N         = length of the path vector (used for plotting)
% nmax      = maximum acceleration capability of the interceptor,
%             altitude dependent
% path      = returned time history of the optimal path
% psi       = initial interceptor heading angle
% psidot    = initial rate of change of interceptor heading angle
% psif      = final interceptor heading angle, calculated from final
%             conditions estimate
% psit      = target heading angle
% Py,Pz     = penalty functions on the y and z acceleration
% q         = variable, counting trajectory updates during intercept
% qq            = variable, counting the number of iterations during
optimization
% range     = estimate of distance between target and interceptor
% state     = state of the interceptor missile, sent from SMGuidance.m
%             [px;py;pz;V;th;psi;Vdot;thetadot;psidot];
% states    = array of the values of all processes in SMGuidance.m
% target    = state of the rocket, sent from SMGuidance.m at i+w,
%             synchronizing times
% tau_f     = value of the virtual arc
% tgo       = time to go to intercept
% th        = initial interceptor flight path angle
% thdot     = initial rate of change of interceptor flight path angle
% thf       = final interceptor flight path angle
% tht       = target flight path angle
% tic..toc  = MATLAB function to track run time
% trys      = vector of optimal path and derivative values
% V         = initial interceptor velocity
% V_f       = final interceptor velocity
% Vave      = average interceptor velocity
% Vdot      = initial interceptor acceleration
% x0        = initial inteceptor position
% xd0       = initial interceptor velocity
% xdd0      = initial inteceptor acceleration
% xdf       = final inteceptor position
% xdt       = current target velocity
% xf        = final inteceptor acceleration
% xmult     = ratio value (used for plotting)
% xt        = current target position
% ymult     = ratio value (used for plotting)
% zmult     = ratio value (used for plotting)

global Re alphag path              % shared by SMFlight3, SMGuidance &
SMTrajectory
global q states                 % shared by SMFlight3 & SMGuidance
global qq thdot psidot tgo costs trys            % ... by SMGuidance &
SMTrajectory

%% Counting trajectory updates during intercept
q=q+1;

%% Initializing Interceptor (all states)
x0=state(1:3);
V=state(4);
th=state(5);
psi=state(6);
Vdot=state(7);
thdot=state(8);
psidot=state(9);
xd0  = [V*cos(th)*cos(psi);
        V*cos(th)*sin(psi);
        V*sin(th)];
xdd0        =        [Vdot*cos(th)*cos(psi)-V*cos(th)*sin(psi)*psidot-
V*sin(th)*cos(psi)*thdot;
        Vdot*cos(th)*sin(psi)+V*cos(th)*cos(psi)*psidot-
V*sin(th)*sin(psi)*thdot;
```

```
        Vdot*sin(th)+V*cos(th)*thdot];

%% Initializing BM (up to the second-order derivatives)
xt  =target(1:3);
xdt =target(4:6);
xddt=target(7:9);

%% Estimating time-to-go
tgo1=100; delta=5;
vmaxhyp=2600;
while delta>1
    if tgo1>20
        V_f=vmaxhyp-10*(tgo1-20);
        Vave=(20*vmaxhyp/2+(tgo1-20)*(vmaxhyp+V_f)/2)/tgo1;
    else
        V_f=tgo1*vmaxhyp/20;
        Vave=V_f/2;
    end
    xf=xt+xdt*tgo1;
    tgo2=sqrt((xf(1)-x0(1))^2+(xf(2)-x0(2))^2+(xf(3)-x0(3))^2)...
         /(norm(xdt)+Vave);
    delta=abs(tgo2-tgo1);
    tgo1=(tgo1+tgo2)/2;
end
tgo=tgo1;

%% Initializing optimization
range=sqrt((xf(1)-x0(1))^2+((xf(2)-x0(2)))^2+((xf(3)-x0(3)))^2);
%fprintf('Trajectory update # %2.0f \n',q)
fprintf('\nSlant range to target:           %5.1fkm \n',range/10^3)
tau_f =0.00045*range-1000*(q-1);   % guess on tau_f
fprintf('Guess on the virtual arc length: %5.1f \n',tau_f)
fprintf('Guess on the time-to-go:         %5.1fkm \n',tgo)
    predicted_xdt=xddt*tgo;
tht =atan2(predicted_xdt(3),norm(predicted_xdt(1:2)));
psit=atan2(predicted_xdt(2),predicted_xdt(1));
thf   =0;%-tht;                 % guess on thf
psif  =psi;%psit+pi;            % guess on psif
free  =[tau_f;thf;psif;.1;1;1;-0.001;-0.001];
BC=[x0;xd0;xdd0;time;xt;xdt;xddt];

%% Searching for the minimum performance index
qq=0;   % counting iterations to converge
tic
options=optimset('MaxIter',100,'Tolfun',1,'TolX',1);
best    =    fminsearch(@(x)    SMTrajectory(x,BC),free,options);       %
Optimization
tcpu=toc;
fprintf('\nIt took %6.0f interations to converge\n',qq)
fprintf('Elapsed time is %6.1f seconds\n',tcpu),
fprintf('Combined performance index is %5.1f\n',costs(end,1))
fprintf('        including:  tau_f=%5.1f,  t2go=%5.1fs,  ImpAngle=%4.1f
off,\n',...

costs(end,2),costs(end,3),180/pi*costs(end,4))
```

```
fprintf('                                                 Pny=%5.1f   and
Pnz=%5.1f\n',costs(end,5),costs(end,6))
%fprintf('                                                Dtgo=%5.1f   and
Altfine=%5.1f\n',costs(end,7),costs(end,8))

tau_f=best(1);
thf  =best(2);
psif =best(3);
[best(4);best(5);best(6);best(7);best(8)];

V_f=path(end,5);
xf=path(end,2:4);
fprintf(['Impact occurs at Altitude of %4.1fkm, Northing=%4.1fkm, and
'...
                              'Westing=%4.1fkm\n'],[xf(3)-Re       xf(1)
xf(2)]/10^3)
%fprintf('with interceptor''s speed of %4.1f km/s\n',V_f/10^3)

xdf=[V_f*cos(thf)*cos(psif);
     V_f*cos(thf)*sin(psif);
     V_f*sin(thf)];

%% Plotting results
%{
xmult=20000/(norm(xd0)+norm(xdt));
ymult=20000/(norm(xd0)+norm(xdt));
zmult=20000/(norm(xd0)+norm(xdt));
nmax=40+(40-10)/(0-50000)*(path(:,4)-Re);

figure('Name','Bird-eye view')              % Bird-eye view
plot3(path(:,2)/10^3,path(:,3)/10^3,(path(:,4)-Re)/10^3,'-
.b','Linewidth',2)
hold on, grid
plot3(10^-3*[x0(1)-xmult*xd0(1);x0(1)+xmult*xd0(1)],...
      10^-3*[x0(2)-ymult*xd0(2);x0(2)+ymult*xd0(2)],...
      10^-3*[x0(3)-Re-zmult*xd0(3);x0(3)-
Re+zmult*xd0(3)],'c','Linewidth',2)
plot3(xf(1)/10^3,xf(2)/10^3,(xf(3)-Re)/10^3,'pr','Linewidth',2)
plot3(10^-3*[xf(1)-xmult*xdt(1);xf(1)+xmult*xdt(1)],...
      10^-3*[xf(2)-ymult*xdt(2);xf(2)+ymult*xdt(2)],...
      10^-3*[xf(3)-Re-zmult*xdt(3);xf(3)-
Re+zmult*xdt(3)],'r','Linewidth',2)
plot3(x0(1)/10^3,x0(2)/10^3,(x0(3)-Re)/10^3,'*b','Linewidth',5)
plot3((x0(1)+xmult*xd0(1))/10^3,(x0(2)+xmult*xd0(2))/10^3,...
                                (x0(3)-
Re+xmult*xd0(3))/10^3,'^c','linewidth',2)
plot3((xf(1)+xmult*xdt(1))/10^3,(xf(2)+xmult*xdt(2))/10^3,...
                                (xf(3)-
Re+xmult*xdt(3))/10^3,'^r','linewidth',2)
hl=legend('Intercept trajectory','Interceptor''s velocity vector',...
       'Impact       point','BM''s       velocity       vector       at
intercept','Location','Best');
set(hl,'FontSize',8);
xlabel('Northing   (km)'),   ylabel('Westing   (km)'),   zlabel('Altitude
(km)')
%title('Interception Geometery','Fontsize',10)
view(-102,8)
axis equal

figure('Name','Combined PI and Virtual arc')     % Performance index &
Virtual arc
subplot(211)
semilogy(costs(:,1)/costs(1,1),'h-.'), grid
xlabel('Iteration'), ylabel('Relative PI (PI_i/PI_1')
xlim([1 qq]), axis 'auto y' % ylim([0 2])
subplot(212)
plot(costs(:,2),'h-.'), grid
xlabel('Iteration'), ylabel('Length of virtual arc, \it\tau_f')
xlim([1 qq])

figure('Name','Delta Time-to-go and Altitude violation') % Dtgo &
Altitude fine
subplot(211)
plot(costs(:,7),'h-.'), grid
xlim([1 qq])
xlabel('Iteration'), ylabel('\Delta \itt_{go} \rm(s)')
subplot(212)
plot(costs(:,8)/10^3,'h-.'), grid
xlabel('Iteration'), ylabel('Alt. violation, (km)')
xlim([1 qq])

figure('Name','Impact angle and Time-to-go')     % Impact angle & Time-
to-go
subplot(211)
plot(real(180/pi*acos(costs(:,4))),'h-.'), grid
axis([1 qq 60 90])
xlabel('Iteration'), ylabel('Impact angle (^o)')
subplot(212)
plot(costs(:,3),'h-.'), grid
xlabel('Iteration'), ylabel('Time-to-go, \itt_{go} \rm(s)')
xlim([1 qq])

figure('Name','G-load factors')                    % G-load constraints
subplot(211)
plot(path(:,1),path(:,9),'-b.'), grid
hold on
plot(path(:,1),path(:,10),'--g.')
plot(path(:,1),nmax(:),'r','Linewidth',2)
plot(path(:,1),-nmax(:),'r','Linewidth',2)
hl=legend('n_y','n_z','Dynamic constraints',2);
set(hl,'FontSize',8);
xlabel('Time (s)'), ylabel('Load factor (g)')
subplot(212)
plot(costs(:,5)/10^7,'-b.','Linewidth',2), grid
hold on
plot(costs(:,6)/10^7,'--g.','Linewidth',2)
xlabel('Iteration'), ylabel('Relative penalty')
hl=legend('n_y penalty','n_z penalty','Location','Best');
set(hl,'FontSize',8);
```

```
xlim([1 qq])

figure('Name','Lambda and Tau profile')          % Lambda and tau
subplot(211)
plot(path(:,1),path(:,end),'.'), grid
xlabel('Time (s)'), ylabel('\it\lambda')
subplot(212)
plot(path(:,1),path(:,end-1),'.'), grid
xlabel('Time (s)'), ylabel('\it\tau')

figure('Name','Speed and Angle of attack profile') % SM speed and Angle
of attack
subplot(211)
plot(path(:,1),path(:,5),'.'); grid
xlabel('Time (s)'), ylabel('Speed, V (m/s)')
subplot(212)
plot(path(:,1),alphag,'.'), grid
xlabel('Time (s)'), ylabel('Angle of attack (^o)')

figure('Name','Euler angles profile')            % SM Euler angles
subplot(211)
plot(path(:,1),path(:,6)*180/pi,'.','Linewidth',2), grid
hold on
plot(path(end,1),thf*180/pi,'ro')
ylim([-30 90])
xlabel('Time (s)'), ylabel('\theta (^o)')
subplot(212)
plot(path(:,1),path(:,7)*180/pi,'g.','Linewidth',2), grid
hold on
plot(path(end,1),psif*180/pi,'ro')
ylim([-180 180])
xlabel('Time (s)'), ylabel('\psi (^o)')
%}

%% Creating results structure
states{q,1}=path;
states{q,2}=BC;
states{q,3}=free;
states{q,4}=best;
states{q,5}=costs;
return
```

### 2. SMGuidanceCost.m

```
function [cost,J,Py,Pz]=SMGuidanceCost(free,const)
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006

% This function calculates the cost of the proposed trajectory returned
% from the SMTrajectory.m function based on the optimization parameters
% and penalty parameters defined herein.  This is a sub-funtion of the
% SMGuidance.m function's fminsearch.  This cost value is used to
% determine whether the proposed trajectory is optimal.  The trajectory
% that returns the minimum value of J is the optimal function.

%% Variable List
% calccost  = a global variable of the value of the J function (used
%             for plotting)
% const     = variables that fminsearch cannot modify, including system
%             contraints, specifically [x0;xd0;xdd0;time;xt;xdt;init]
% cost      = cost function value returned from SMGuidanceCost.m
function
% costs     = vector of values of the cost variables at each iteration
% dist      = cumulative distance travelled
% free      = variables that fminsearch can modify, specifically
%            [tau;tgo;thf;psif]
% init      = vector of initial estimates
% J         = vector of cost function variable values
% N         = length of the path vector (used for plotting)
% nmax      = maximum acceleration capability of the interceptor,
%             altitude dependent
% nx        = axial acceleration command, body frame x
% ny        = axial acceleration command, body frame y
% nz        = axial acceleration command, body frame z
% path      = returned time history of the optimal path, specifically
%            [time' X(1:3,:)' V' th' psi' Vdot' thdot' psidot' nx' ny'
nz']
% psi       = initial interceptor heading angle
% psidot    = initial rate of change of interceptor heading angle
% psif      = final interceptor heading angle, calculated from final
%             conditions estimate
% Py        = penalty function on the y acceleration
% Pz        = penalty function on the z acceleration
% qq        = counting variable
% t         = current time
% tau_f     = value of the virtual arc
% tgo;      = time to go to intercept
% th        = initial interceptor flight path angle
% thdot     = initial rate of change of interceptor flight path angle
% thf       = final interceptor flight path angle
% time      = optimal path time history
% V         = initial interceptor velocity
% V_f       = final interceptor velocity
% Vdot      = initial interceptor acceleration
% X         = the optimal path time history in cartesian coordinates
% x0        = initial inteceptor position
% xd0       = initial interceptor velocity
% xdd0      = initial inteceptor acceleration
% xdf       = final inteceptor velocity
% xdt       = current target velocity
% xt        = current target position

[path]=SMTrajectory(free,const);

global calccost costs q qq tgo trys
```

```
% Initialize Variables
qq=qq+1;
dist=0;
Re=6.378137e6;

%% Identify Variables
time=path(:,1);
X=path(:,2:4);
V=path(:,5);
th=path(:,6);
psi=path(:,7);
Vdot=path(:,8);
thdot=path(:,9);
psidot=path(:,10);
nx=path(:,11);
ny=path(:,12);
nz=path(:,13);
N=length(path(:,1));

tau_f=free(1);
tgo=free(2);
thf=free(3);
psif=free(4);

x0=const(1:3);
xd0=const(4:6);
xdd0=const(7:9);
t=const(10);
xt=const(11:13);
xdt=const(14:16);
init=const(17:19);

V_f=path(N,5);
xdf=[V_f*cos(thf)*cos(psif);
     V_f*cos(thf)*sin(psif);
     V_f*sin(thf)];
tgo=path(N,1);

for i=2:1:N
    dist=dist+abs(norm([X(i,1)-X(i-1,1);X(i,2)-X(i-1,2);X(i,3)-X(i-
1,3);]));
end
nmax=40+(40-10)/(0-50000)*(path(:,4)-Re);

%% Calulate Cost of the chosen trajectory
J=[ tau_f;
    tgo;
    100*abs(dot(xdf,xdt)/norm(xdf)/norm(xdt))];
Py=sum(max(0,abs(ny)-nmax).^2);
Pz=sum(max(0,abs(nz)-nmax).^2);

cost=0.33*ones(1,3)*J+norm([Py;Pz]);
costs(qq,1:6)=[cost;J;Py;Pz;];
calccost=cost;
```

```
return
```

### 3. SMTrajectory.m

```
function cost=SMTrajectory(free,BC)
% This function computes a candidate trajectory and associated cost
based on
% the vector of varied parameters "free" and boundary conditions "BC".

% This is a sub-funtion of the SMGuidance.m function's fminsearch.
% This function creates a 7th order set of equations and evaluates that
set at
% the boundary conditions supplied by the inputs. It then calculates
the time
% history of all the flight vehicle variables, including controls and
reactions,
% necessary to develop that flight path. A plot command set at the end
of this
% function will plot a chart of the iterations at the end of run if
desired.
% Finally, this function calculates the cost of the candidate
trajectory
% combining the value of the performance index and penalties. This cost
value is
% used to determine whether the proposed trajectory is optimal. (The
trajectory
% that returns the minimum value of the cost is the sub-optimal one.)

% O.Yakimenko, Naval Postgraduate School, November 2009

%% List of variables
% A,Ax,Axp,Axpp,Axppp    = cell matrices of coefficients of a candidate
reference
%                           trajectory and their derivatives wrt virtual
arc
% BC                        = boundary conditions, specifically
[x0;xd0;xdd0;time;xt;xdt;xddt]
% Cx,Cxp,Cxpp,Cxppp         = coefficients of a candidate reference
trajectory and
%                          their derivatives wrt virtual arc
% dtau      = tau step value
% dtime     = time step value
% free      = variable parameters, specifically [tau;thf;psif]
% g         = gravitational force
% L         = lambda, virtual speed
% Lp        = first-order derivative of lambda wrt to virtual arc
% nmax      = maximum acceleration capability of the interceptor,
%             altitude dependent
% nX,nXp,nXpp,nXppp         = norm of reference trajectory and its
```

```
derivatives
% nx        = axial acceleration command, body frame x
% ny        = axial acceleration command, body frame y
% nz        = axial acceleration command, body frame z
% path      = returned time history of the optimal path, specifically
%             [time' X(1:3,:)' V' th' psi' nx' ny' nz' tau' L']
% psi       = interceptor heading angle
% psidot    = rate of change of interceptor heading angle
% qq        = variable counting the number of iterations
% t         = global current time
% tau       = virtual arc
% tau_f     = length of the virtual arc
% tgo       = time to go to intercept
% th        = interceptor flight path angle
% thdot     = rate of change of interceptor flight path angle
%  thp           =  first-order  derivative  of  flight  path  angle  wrt  to
virtual arc
% time      = optimal path time history time=[0;tgo]
%  trys        = collection  of  norms  [X  nX  Xp  nXp  Xpp  nXppp]  (used  for
plotting)
% V         = velocity
% Vdot      = acceleration
% Vp        = irst-order derivative of velocity wrt to virtual arc
% X,Xp,Xpp,Xppp          = reference trajectory and its derivatives wrt
tau
% x0,xf     = initial and final inteceptor position
% xd0,xdf   = initial and final interceptor velocity
% xdd0,xddf = initial and final inteceptor acceleration
%  xp,xpp,xppp               =  boundary  conditions  (at  0  and  f)  in  the
virtual domain
% xt,xdt,xddt            = target position, velocity and acceleration at
time=0

global Re alphag path              % shared by SMFlight3, SMGuidance &
SMTrajectory
global Pos_BR Pos_SM Npol Npt kpolar weight90    % ... by SMFlight3 &
SMTrajectory
global qq thdot psidot tgo costs trys           % ... by SMGuidance &
SMTrajectory

%% Counting iterations to converge
qq=qq+1;
tgoold=tgo;

%% Assigning variables
tau_f=free(1);
thf  =free(2);
psif =free(3);
t=BC(10);  % current time in the SM frame to compute it's stage (thrust
and drug)

x0=BC(1:3);
xd0=BC(4:6);
xdd0=BC(7:9);
V(1)=norm(xd0);
```

```
Vdoti=norm(xdd0);

L(1)=1;%V(1);
Lpi=0;%Vdoti/L(1)

xt=BC(11:13);   % Target's coordinates at the moment of detection
xdt=BC(14:16);  % Target's velocities at the moment of detection
xddt=BC(17:19); % Target's accelerations at the moment of detection

V(Npt)=2600-1/3*(tgo-20);   % SM velocity estimate at impact
Vdotf=-0.3;%-5.9578;        % SM acceleartion estimate at impact

L(Npt)=1;%V(Npt);
Lpf=0;%Vdotf/L(Npt);

xf=xt+xdt*tgo+0.5*xddt*tgo^2;
xdf=[V(Npt)*cos(thf)*cos(psif);
     V(Npt)*cos(thf)*sin(psif);
     V(Npt)*sin(thf)];

thdotf=free(7); psidotf=free(8);
xddf=[Vdotf*cos(thf)*cos(psif)-V(Npt)*cos(thf)*sin(psif)*psidotf-...
      V(Npt)*sin(thf)*cos(psif)*thdotf;
      Vdotf*cos(thf)*sin(psif)+V(Npt)*cos(thf)*cos(psif)*psidotf-...
      V(Npt)*sin(thf)*sin(psif)*thdotf;
      Vdotf*sin(thf)+V(Npt)*cos(thf)*thdotf];

%% Converting boundary conditions ito the virtual domain
xp0=xd0/L(1);
xpp0=(xdd0-xd0*Lpi)/L(1)^2;
xppp0=[free(4);free(5);free(6)];

xpf=xdf/L(Npt);
xppf=(xddf-xdf*Lpf)/L(Npt)^2;
xpppf=[0;0;0];

%% Calculating polynomials' coefficients and reference trajectories
    dtau =tau_f/(Npt-1);
    tau  =linspace(0,tau_f,Npt);
for i=1:3
A{i}=[x0(i);
      xp0(i);
      xpp0(i);
      xppp0(i);
      (-16*xppp0(i)-4*xpppf(i))/tau_f+(-
120*xpp0(i)+60*xppf(i))/tau_f^2+...
                  (-360*xpf(i)-480*xp0(i))/tau_f^3+(840*xf(i)-
840*x0(i))/tau_f^4;
      (60*xppp0(i)+30*xpppf(i))/tau_f^2+(600*xpp0(i)-
420*xppf(i))/tau_f^3+...
               (2340*xpf(i)+2700*xp0(i))/tau_f^4+(5040*x0(i)-
5040*xf(i))/tau_f^5;
      (-80*xppp0(i)-60*xpppf(i))/tau_f^3+(780*xppf(i)-
900*xpp0(i))/tau_f^4+...
```

```
             (-4080*xpf(i)-4320*xp0(i))/tau_f^5+(-
8400*x0(i)+8400*xf(i))/tau_f^6;
      (35*xppp0(i)+35*xpppf(i))/tau_f^4+(420*xpp0(i)-
420*xppf(i))/tau_f^5+...
              (2100*xpf(i)+2100*xp0(i))/tau_f^6+(4200*x0(i)-
4200*xf(i))/tau_f^7];
    Ax{i}    =diag([  1,     1,     1/2,  1/6,   1/24,  1/60,  1/120,  1/210
])*A{i};
    Axp{i}   =diag([  0,     1,     1,     1/2,   1/6,   1/12,  1/20,   1/30
])*A{i};
    Axpp{i}  =diag([  0,     0,     1,     1,      1/2,   1/3,   1/4,    1/5
])*A{i};
    Axppp{i}=diag([ 0,    0,    0,    1,     1,     1,     1,      1  ])*A{i};
    Cx(i,:)    =Ax{i}([Npol:-1:1]);
    Cxp(i,:)   =Axp{i}([Npol:-1:2]);
    Cxpp(i,:) =Axpp{i}([Npol:-1:3]);
    Cxppp(i,:)=Axppp{i}([Npol:-1:4]);
    X(i,:)    =polyval(Cx(i,:),tau);
    Xp(i,:)   =polyval(Cxp(i,:),tau);
    Xpp(i,:) =polyval(Cxpp(i,:),tau);
    Xppp(i,:)=polyval(Cxppp(i,:),tau);
end

%% Computing the states
xp12=Xp(1,:).^2+Xp(2,:).^2;
th  =atan2(Xp(3,:),sqrt(xp12));
psi =atan2(Xp(2,:),Xp(1,:));
thp                                                  =(Xpp(3,:).*xp12-
Xp(3,:).*(Xp(1,:).*Xpp(1,:)+Xp(2,:).*Xpp(2,:)))./...

sqrt(xp12)./(xp12+Xp(3,:).^2);
psip =(Xp(1,:).*Xpp(2,:)-Xpp(1,:).*Xp(2,:))./xp12;

time(1)=t;
g      = 3.986004418e14/norm(X(1:3,1))^2;
nx(1)  = Vdoti/g+sin(th(1));
ny(1)  = V(1)/g*cos(th(1))*psidot;
nz(1)  = V(1)/g*thdot+cos(th(1));

[ro,press,temp]=STatmos(norm(X(:,1))-Re);
[m_i,Sref]=SMParams3(time(1));
alphag(1)=180/pi*sqrt(ny(1)^2+nz(1)^2)*m_i*g/(ro*V(1)^2*Sref/2)/13;

for j=2:Npt;
    g=3.986004418e14/norm(X(1:3,j))^2;
        if  norm(X(:,j))-Re<86000,  [ro,press,temp]=STatmos(norm(X(:,j))-
Re);
        else                             [ro,press,temp]=STatmos(86000);
end
    MV=V(j-1)/sqrt(1.402*287.053*temp);
    CDtable=SMDrag(MV);
        if time(j-1)<20    Thrust=95300;   CD0=CDtable(1);
        else                         Thrust=0;           CD0=CDtable(2);
end
    [m_i,Sref]=SMParams3(time(j-1));
```

```
            CL=(2*sqrt(ny(j-1)^2+nz(j-1)^2)*m_i*g)/(ro*V(j-1)^2*Sref);
            CD = CD0 + kpolar*CL^2;
    Drag=ro*V(j-1)^2*CD*Sref/2;
    nx(j)=(Thrust-Drag)/m_i/g;

    V(j)=V(j-1)+g*(nx(j-1)-sin(th(j-1)))/L(j-1)*dtau;

    ddist=sqrt((X(1,j)-X(1,j-1))^2+(X(2,j)-X(2,j-1))^2+(X(3,j)-X(3,j-
1))^2);
    dtime=2*ddist/(V(j)+V(j-1));
    L(j)=dtau/dtime;

    ny(j)=V(j)/g*cos(th(j))*psip(j)*L(j);
    nz(j)=V(j)/g*thp(j)*L(j)+cos(th(j));
    alphag(j)=180/pi*sqrt(ny(j)^2+nz(j)^2)*m_i*g/(ro*V(j)^2*Sref/2)/13;

    time(j)=time(j-1)+dtime;
end

tgo=time(end)-time(1);

for i=1:Npt
    nX(i)=norm(X(1:3,i));
    nXp(i)=norm(Xp(1:3,i));
    nXpp(i)=norm(Xpp(1:3,i));
end

trys=[X' nX' Xp' nXp' Xpp' nXpp'];
path=[time' X(1:3,:)' V' th' psi' nx' ny' nz' tau' L'];

%% Computing cost and penalties
V_f=V(end);
xdt=BC(14:16)+BC(17:19)*tgo;
xdf=[V_f*cos(thf)*cos(psif);
     V_f*cos(thf)*sin(psif);
     V_f*sin(thf)];

nmax=40+(40-10)/(0-50000)*(X(3,:)-Re);
if nmax<1, nmax=1; end

J=[tgo;
   abs(dot(xdf,xdt))/norm(xdf)/norm(xdt)];
Py=max([0,abs(ny)-nmax]);
Pz=max([0,abs(nz)-nmax]);
Dtgo=tgoold-tgo;
Altfine=max([0,X(3,end)-Re-60000]).^2+min([0,X(3,end)-Re-9000]).^2;
cost=norm([1,weight90]*J)+10*(Py^2+Pz^2)+100*Dtgo^2+0.1*Altfine;
costs(qq,:)=[cost;tau_f;J;Py;Pz;Dtgo;Altfine];

%% Animating iterations


figure(100),% set(gcf,'Color','w');
```

```
subplot(3,6,[1 2 7 8])

plot3(Pos_BR(1:300,1)/10^3,Pos_BR(1:300,2)/10^3,(Pos_BR(1:300,3)-
Re)/10^3,...
                                   '-.r','LineWidth',2)
hold on, ylim([0 60]); %axis equal
plot3(Pos_BR(140,1)/10^3,Pos_BR(140,2)/10^3,(Pos_BR(140,3)-Re)/10^3,...
                                   '^g','LineWidth',2)
plot3(Pos_SM(1:19,1)/10^3,Pos_SM(1:19,2)/10^3,(Pos_SM(1:19,3)-
Re)/10^3,...
                                   '.k','LineWidth',2)

plot3(X(1,:)/10^3,X(2,:)/10^3,(X(3,:)-Re)/10^3,'Linewidth',3); grid on
plot3(xf(1)/10^3,xf(2)/10^3,(xf(3)-Re)/10^3,'pk','MarkerSize',11)
plot3(X(1,end)/10^3,X(2,end)/10^3,(X(3,end)-
Re)/10^3,'pr','MarkerSize',9);
view(-102,8)
hl=legend('BM trajectory','BM detection','Unguided ascend',...
       'Guided  flight','Predicted  intercept  point','Actual  intercept
point',...
       'Location','North'); set(hl,'FontSize',7)
hold off
%{
axis([0 1.1e5 -1e4 1.1e5 0 7e4]/10^3)
xlabel('Northing,  x  (km)'),ylabel('Westing,  y  (km)'),zlabel('Altitude
(km)')
        subplot(3,6,[13 14])
        plot(path(:,1),path(:,9),'--b','Linewidth',2)
        hold on; grid on
        plot(path(:,1),path(:,10),'-.g','Linewidth',2)
        plot(path(:,1),nmax(:),'r','Linewidth',2)
        plot(path(:,1),-nmax(:),'r','Linewidth',2)
        axis([10 time(Npt) -45 45]);
        xlabel('Time (s)'), ylabel('Load Factor (g)')
        leg2=legend('n_y','n_z','Dynamic
constraints','Location','Best');
        set(leg2,'FontSize',7)
        hold off
      subplot(3,6,3)
      plot(time,X(1,:)/10^3,'Linewidth',2); grid on
      axis([10 time(Npt) 1e4/10^3 1.1e5/10^3])
      title('x_1 (km)')
    subplot(3,6,9)
    plot(time,X(2,:)/10^3,'Linewidth',2); grid on
    axis([10 time(Npt) -1e4/10^3 1.1e5/10^3])
    title('x_2 (km)')
  subplot(3,6,15)
  plot(time,(X(3,:)-Re)/10^3,'Linewidth',2); grid on
  axis([10 time(Npt) 0 7e4/10^3])
  title('x_3 (km)'), xlabel('Time (s)')
subplot(3,6,4)
plot(time,Xp(1,:)/10^3,'Linewidth',2); grid on
axis([10 time(Npt) -45 45]); axis 'auto y'
title('x_1'' /10^3')
  subplot(3,6,10)
  plot(time,Xp(2,:)/10^3,'Linewidth',2); grid on
  axis([10 time(Npt) -45 45]); axis 'auto y'
  title('x_2'' /10^3')
    subplot(3,6,16)
    plot(time,Xp(3,:)/10^3,'Linewidth',2); grid on
    axis([10 time(Npt) -45 45]); axis 'auto y'
    title('x_3'' /10^3'), xlabel('Time (s)')
      subplot(3,6,5)
      plot(time,Xpp(1,:)/10^2,'Linewidth',2); grid on
      axis([10 time(Npt) -45 45]); axis 'auto y'
      title('x_1'''' /10^2')
        subplot(3,6,11)
        plot(time,Xpp(2,:)/10^2,'Linewidth',2); grid on
        axis([10 time(Npt) -45 45]); axis 'auto y'
        title('x_2'''' /10^2')
      subplot(3,6,17)
      plot(time,Xpp(3,:)/10^2,'Linewidth',2); grid on
      axis([10 time(Npt) -45 45]); axis 'auto y'
      title('x_3'''' /10^2'), xlabel('Time (s)')
    subplot(3,6,6)
    hold on
    plot(qq,time(Npt)-time(1),'+c','Linewidth',1); grid on
    hold off
    axis([1 200 40 60]); axis 'auto y'
    title('Time-to-go (s)')
  subplot(3,6,[12 18])
  hold on
  plot(qq,cost,'+m','Linewidth',1); grid on
  axis([1 200 0 100]); axis 'auto y'
  hold off
  title('Performance Index'), xlabel('Iteration')
%}
return
```

## D. COMMON FUNCTION - STANDARD ATMOSPHERE

### 1. STatmos.m

```
function [Density, Pressure, Temperature]=STatmos(alt)
% Calculation of the 1976 standard atmosphere up to 86 km
% Code source:  http://www.pdas.com/atmos.htm
% Run ezplot('STatmos',[0,86000]) to see the plot of density vs
altitude
%
% Author: Yakimenko, Oleg A.
% Date:   September, 27 2005
% E-mail: oayakime@nps.edu
%
alt=alt/1000;               % Convert altitude from m to km
%% --- Initialize values for 1976 atmosphere
```

```
REARTH=6369.0;              % Earth radius (km), depends on Latitude
GMR=34.163195;              % Gas Constant
htab=[0.0, 11.0, 20.0, 32.0, 47.0, 51.0, 71.0, 84.852]; % Geometric alt
ttab=[288.15, 216.65, 216.65, 228.65, 270.65,...        % Temperature
      270.65, 214.65, 186.946];
ptab=[1.0, 2.233611E-1, 5.403295E-2, 8.5666784E-3,...   % Relative pres
      1.0945601E-3, 6.6063531E-4, 3.9046834E-5, 3.68501E-6];
gtab=[-6.5, 0.0, 1.0, 2.8, 0.0, -2.8, -2.0, 0.0];       % Temp gradient
P0=101325.0; Ro0=1.225;

%%--- Convert geometric to geopotential altitude
      if alt>250
          alt = 100
      end

      h=alt*REARTH/(alt+REARTH);

%% --- Binary search for altitude interval
      i = 1;
      j = 8;

while j > i+1
      k=fix((i+j)/2);
      if h<htab(k);
        j = k;
      else
        i = k;
      end
end

%% --- Calculate local temperature
      tgrad  = gtab(i);
      tbase  = ttab(i);
      deltah = h - htab(i);
      tlocal = tbase + tgrad*deltah;
      theta  = tlocal/ttab(1);

%% --- Calculate local pressure
if (tgrad == 0.0)
  delta = ptab(i)*exp(-GMR*deltah/tbase);        % Isothermal layers
else
  delta = ptab(i)*(tbase/tlocal)^(GMR/tgrad);    % Non-isothermal
layers
end

%% --- Calculate local density
sigma = delta/theta;

%% --- Current atmosphere parameters corresponding to Altitude alt
Temperature=tlocal; Density=Ro0*sigma; Pressure=P0*delta;
return
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX E: DETAILED DESCRIPTION OF TRAJECTORY SHAPING GUIDANCE (REPRODUCED FROM [10])

A method that overcomes fatal characteristics of modern guidance laws is the key driving factor in the development of this advanced guidance law. The guidance law will determine the near-optimal flight path from the interceptor position to a predicted target position for the interceptor to follow to intercept, and then derive the set of control commands necessary to execute that flight. This section will describe a method for deriving that trajectory using calculus of variations based on three cues: high-order polynomials as a reference function for the flight path, a preset thrust history as one of the controls, and a few optimization parameters. The trajectory optimization problem is then converted into a nonlinear programming problem and solved numerically.

## A. PROBLEM STATEMENT

Among all admissible trajectories,

$$\begin{gathered}\vec{z}(t)=\{z_1(t),z_2(t),\dots,z_r(t)\}^T\in S\\ S=\left\{\vec{z}(t)\in Z^r\subset E^r\right\},\ t\in\left[t_0,t_f\right]\end{gathered}\tag{4.A.1}$$

that satisfy:

1. The system of differential equations (dynamic constraints):

$$\dot{z}_i=f_i(t,\vec{z},\vec{u},\vec{a}),\ i=\overline{1,r}\tag{4.A.2}$$

where the vector of controls is $\vec{u}(t)=\left\{u_1(t),u_2(t),\dots,u_m(t)\right\}^T,\ m<r,\ \vec{u}\in U^m\subset E^m$ and the vector of missile parameters is $\vec{a}=(a_1,a_2,\dots,a_p),\ \vec{a}\in A^p\subset E^p$;

2. The initial conditions:

$$\vec{z}(t_0)\in S_0,\quad S_0=\left\{\vec{z}_0\in Z^r\subset E^r\right\}\tag{4.A.3}$$

$$\vec{u}(t_0)\in R_0,\quad R_0=\left\{\vec{u}_0\in U^m\subset E^m\right\}\tag{4.A.4}$$

and the final conditions:

$$\vec{z}\left(t_f\right)\in S_f,\ S_f=\left\{\vec{z}_f\in Z^r\subset E^r;\ G_j\left(\vec{z}\left(t_f\right)\right)=0,\ j=\overline{1,l}\right\}\tag{4.A.5}$$

$$\vec{u}(t_f)\in R_f,\ R_f=\left\{\vec{u}_f\in U^m\subset E^m\right\}\tag{4.A.6}$$

3. The constraints imposed on the state space

$$\vec{\eta}(t,\vec{z}) = \left\{\eta_1(t,\vec{z}), \eta_2(t,\vec{z}), ..., \eta_\mu(t,\vec{z})\right\}^T \geq \vec{0} \tag{4.A.7}$$

on the controls

$$\vec{\xi}(t,\vec{z},\vec{u}) = \left\{\xi_1(t,\vec{z},\vec{u}), \xi_2(t,\vec{z},\vec{u}), ..., \xi_\nu(t,\vec{z},\vec{u})\right\}^T \geq \vec{0} \tag{4.A.8}$$

and on the controls derivatives

$$\vec{\pi}(t,\vec{u},\dot{\vec{u}}) = \left\{\pi_1(t,\vec{u},\dot{\vec{u}}), \pi_2(t,\vec{u},\dot{\vec{u}}), ..., \pi_\sigma(t,\vec{u},\dot{\vec{u}})\right\}^T \geq \vec{0} \tag{4.A.9}$$

Find the optimal trajectory, $\vec{z}_{opt}(t)$, that minimizes the integral function

$$J = K(x_0, x_f) + \int_{t_0}^{t_f} L(t,\vec{z},\vec{u})dt \tag{4.A.10}$$

and the corresponding optimal controls, $\vec{u}_{opt}(t)$, where *K*, *L* are defined functions.

## B. CALCULUS OF VARIATIONS

Calculus of variations deals with functions of functions, termed functional, instead of functions of some variable or variables as in ordinary calculus. Specific interest is in the externals of these functionals - those making the functional attain a maximum or minimum value [Ref 15]. There are two broad categorizations of methods to solve these problems, indirect methods and direct methods.

Indirect methods resolve the problem into a differential equation, usually via the difference of a series of equations of motion and thus solve the general theory of partial differential equations. This method does not assume anything about the solution, leading to a very complete and perfectly precise answer; however, one must integrate the resultant equation to derive the extremals. The precision greatly increases the computational complexity, and therefore the time required to solve, for negligible gain in optimality. Further, these differential equations are difficult to integrate except in the simplest of cases, and nearly impossible to program [Ref 19]. This approach is further complicated by the need to solve the problem in a specified fixed region instead of in the small neighborhood of some point. These difficulties can be overcome by using direct methods, which do not reduce the variational problems to ones involving differential equations.

The fundamental idea of direct methods is to consider a variational problem as a limit problem of the extreme of a function of a finite number of variables that can be solved by numerical methods.  Two basic direct methods are the Rayleigh-Ritz and Galerkin methods, which assume the solution to be an unknown function but of a certain form containing a set of unknown coefficients (themselves functions of the boundary conditions), which are then found by minimization. The practical result is that the problem has been reduced from calculus to algebra, but at the cost of a significant increase in the number of simultaneous equations to be solved.  It is for this reason that little work was done in this field until the advent of computer technology.  The possible solution set is restricted to a smaller space than the original equation because of the initial assumption regarding the solution, but the resultant problem can be programmed and quickly solved using computers; however, the solutions are only an approximation of the original solution.  Therefore these solutions can only be regarded as near-optimal solutions.

Professor Taranenko [Ref 19] first applied the ideas of direct methods and the combination of Ritz and Galerkin methods to the problems of flight dynamics by identifying a reference function for the flight vehicle's motion and velocity

$$x_i = x_{i0} + (x_{if} - x_{i0})\frac{\tau - \tau_0}{\tau_f - \tau_0} + \Phi_i(\tau),\ i = \overline{1,4} \qquad (2.1)$$

where $x_1, x_2, x_3,$ are the Cartesian coordinates for the flight path, $x_4$ is the velocity, and $\Phi_i(\tau)$ is a continuous, unequivocal, differentiable function satisfying the boundary conditions $\Phi_i(\tau_0) = \Phi_i(\tau_f) \equiv 0$.  Any function that satisfies those conditions would be acceptable for use.  Taranenko called $\tau$ a virtual arc. It is this critical variable that allows the separation of the spatial trajectory from the velocity and thus optimize one or the other, or both independently. The specific task determines the appropriate choice of $\tau$, but in general any continuous, monatomic function is acceptable: time, path, energy, etc. The remaining state parameters and flight controls are then determined by solving the inverse problem of flight dynamics.  Rather than starting with the control time histories and integrating them to determine the flight path, this method starts with a flight path and determines the control time histories necessary to create it.

In order to implement a solution to this problem that can be solved in real time, a further restriction must be made. The continuous problem must be discretized to reduce the infinite variational problem to one of optimization of few parameters at numerous sampling points. This allows for the optimization of the planar trajectory of the missile at several points along the path by presetting the state variables and one of the controls' time histories, and then solving the inverse flight dynamics problem.

These methods have all previously been used for off-line optimization of trajectories, but none has yet been applied to the real-time onboard optimization of a missile flight path. Yakimenko detailed a method called a Direct Method for Rapid Prototyping, which he applied to short term spatial trajectories of aircraft maneuvers using fixed boundary points [Ref 19]. The developed program presented here uses similar numerical method to provide a near-optimal spatial trajectory that is completely defined by a few optimization parameters, but as will be discussed shortly, has fluid final boundary conditions.

Though the method artificially limits the possible trajectory variations, it does guarantee the following [Ref 19]:

1. The boundary conditions are satisfied a priori,
2. The control commands are physically realizable and smooth,
3. Only a few variable parameters are used, thus ensuring that the iterative process converges well,
4. The near-optimal solution is very close to the optimal one.

A simple two dimensional variation program of a similar 7th order system will demonstrate how the direct method varies the flight path according to the boundary conditions. The boundary conditions are

$$\begin{array}{cccc} x_{10} = 0 & x_{20} = 0 & x_{1f} = 1 & x_{2f} = 1 \\ x'_{10} = 0.2 & x'_{20} = 1 & x'_{1f} = 0.1 & x'_{2f} = -1 \\ x''_{10} = 0.1 & x''_{20} = 0.1 & x''_{1f} = 0.1 & x''_{2f} = 0.1 \\ x'''_{10} = \mathrm{var} & x'''_{20} = 0.1 & x''''_{1f} = 0.1 & x'''_{2f} = 0.1 \end{array}$$

where the third derivative of the initial $x_1$ condition has been set to vary according to

$$x'''_{10} = \{-0.4; -0.1; 0.2; 0.5\}$$

and the length of the virtual arc varies according to

$$\tau_f = 1, 2, ..., 10$$

The resulting set of paths is shown in figure 26 and the first derivative of the path is shown in figure 27. It is important to note that the first derivative is not "velocity" but is instead the "rate of change of the path". It is proportional but not equal to velocity, specifically because of the virtual variable $\tau$ as discussed previously. Each line represents a different choice of $\tau_f$, showing that by varying that value the length of the path can change drastically. The algorithm developed here will use this technique to derive the optimal flight path.

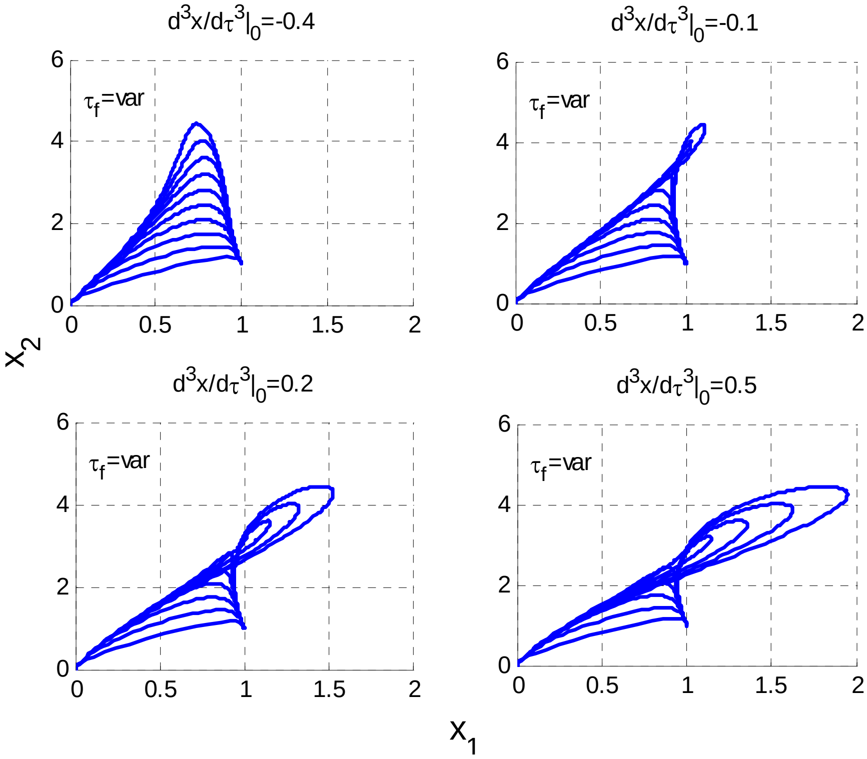


Figure 37. Variation of Path with $\tau_f$ and $x'''_{10}$ (after [Ref 18])

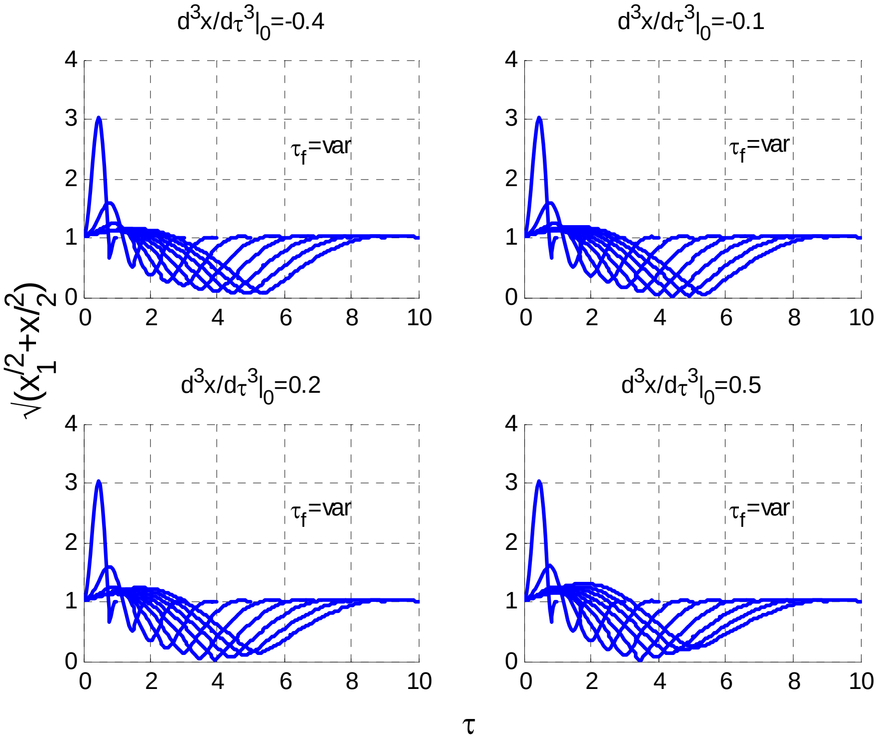


Figure 38. Variation of First Derivative of Path with $\tau_f$ and $x_{10}'''$

## C. PROGRAM DEVELOPMENT

### 1. Boundary Conditions

Using equations (3.2) and (3.3), in addition to the controls $n_x, n_y, n_z$, it is possible to construct the vector of state variables $z = \{x_1, x_2, x_3, V, \theta, \Psi\}^T$ and the vector of controls $u = \{n_x, n_y, n_z\}^T$. The model for thrust, drag, and missile characteristics is the same as previously used in Chapter 3.

The beginning assumption is that the following data is known by the interceptor's onboard computer:

|  | Interceptor | Target |
|---|---|---|
| Body Frame | $x_1(t_0)=x_{10}$<br>$x_2(t_0)=x_{20}$<br>$\left.\begin{array}{l}x_3(t_0)=x_{30}\\ V(t_0)=V_0\\ \Psi(t_0)=\Psi_0\\ \theta(t_0)=\theta_0\end{array}\right\}\left.\begin{array}{l}\dot{x}_{10}\\ \dot{x}_{20}\\ \dot{x}_{30}\\ n_y(t_0)=n_{y0}\\ n_z(t_0)=n_{z0}\end{array}\right\}\begin{array}{l}\ddot{x}_{10}\\ \ddot{x}_{20}\\ \ddot{x}_{30}\end{array}$ | $x_1(t_f)=x_{1f}$<br>$x_2(t_f)=x_{2f}$<br>$\left.\begin{array}{l}x_3(t_f)=x_{3f}\\ V(t_f)=V_f\\ \Psi(t_f)=\Psi_f\\ \theta(t_f)=\theta_f\end{array}\right\}\begin{array}{l}\dot{x}_{1f}\\ \dot{x}_{2f}\\ \dot{x}_{3f}\end{array}$ |
| Earth Centered Inertial | $x_1^{SM}(t),\ x_2^{SM}(t),\ x_3^{SM}(t)$<br>$\dot{x}_1^{SM}(t),\ \dot{x}_2^{SM}(t),\ \dot{x}_3^{SM}(t)$<br>$\ddot{x}_1^{SM}(t),\ \ddot{x}_2^{SM}(t),\ \ddot{x}_3^{SM}(t)$ | $x_1^{BR}(t),\ x_2^{BR}(t),\ x_2^{BR}(t)$<br>$\dot{x}_1^{BR}(t),\ \dot{x}_2^{BR}(t),\ \dot{x}_3^{BR}(t)$ |

Table 9. Interceptor Known Data

The target data will be used to determine the final boundary conditions of the interceptor missile according to the time until intercept, $\Delta t$:

Position:

$$\begin{aligned} x_{1f}^{SM} &= x_{10}^{BR}+\dot{x}_{10}^{BR}\Delta t\\ x_{2f}^{SM} &= x_{20}^{BR}+\dot{x}_{20}^{BR}\Delta t\\ x_{3f}^{SM} &= x_{20}^{BR}+\dot{x}_{20}^{BR}\Delta t \end{aligned} \tag{4.C.1}$$

Heading and Flight Path Angle:

$$\begin{aligned} \theta_f^{SM} &= \frac{\pi}{2}+\theta_f^{BR},\ \text{where } \theta_f^{BR}=\text{arctg}\frac{\dot{x}_3^{BR}}{\sqrt{\dot{x}_1^{BR2}+\dot{x}_2^{BR2}}}\\ \psi_f^{SM} &= \psi_f^{BR},\ \text{where } \psi_f^{BR}=\text{arctg}\frac{\dot{x}_2^{BR}}{\dot{x}_1^{BR}} \end{aligned} \tag{4.C.2}$$

which, when combined with reasonable estimates of the final values of the velocity, *V*, and the time rate of change of velocity, $\dot{V}$, heading ($\dot{\Psi}=0$), and flight path angle ($\dot{\theta}=0$), yields the final conditions:

$$
\begin{aligned}
\dot{x}_{1f}^{SM} &= V_f \cos\theta_f \cos\psi_f \\
\dot{x}_{2f}^{SM} &= V_f \cos\theta_f \sin\psi_f \\
\dot{x}_{3f}^{SM} &= V_f \sin\theta_f
\end{aligned}
\tag{4.C.3}
$$

and

$$
\begin{aligned}
\ddot{x}_{1f}^{SM} &= \dot{V}_f \cos\theta_f \cos\Psi_f - \dot{\theta}_f V_f \sin\theta_f \cos\Psi_f - \dot{\Psi}_f V_f \cos\theta_f \sin\Psi_f \\
\ddot{x}_{2f}^{SM} &= \dot{V}_f \cos\theta_f \sin\Psi_f - \dot{\theta}_f V_f \sin\theta_f \sin\Psi_f + \dot{\Psi}_f V_f \cos\theta_f \cos\Psi_f \\
\ddot{x}_{3f}^{SM} &= \dot{V}_f \sin\theta_f + \dot{\theta}_f V_f \cos\theta_f
\end{aligned}
\tag{4.C.4}
$$

In order to ensure a smooth flight path at the initial and final points, an additional constraint of

$$
\begin{aligned}
\dddot{x}_{1i}^{SM} &= \dddot{x}_{1f}^{SM} = 0 \\
\dddot{x}_{2i}^{SM} &= \dddot{x}_{2f}^{SM} = 0 \\
\dddot{x}_{3i}^{SM} &= \dddot{x}_{3f}^{SM} = 0
\end{aligned}
\tag{4.C.5}
$$

will be imposed on the system at the initial and final conditions.

### 2. Separating and Recombining Space and Time

As discussed previously, in order to independently optimize the spatial trajectory and the velocity, the reference function will be derived as a function of $\tau$. The boundary conditions cannot, therefore, be defined as functions of time derivatives as in equations (4.C.3) – (4.C.5). A connection between the spatial and time domains must therefore be introduced, $\lambda$, which is defined as

$$
\lambda(\tau) = \frac{d\tau}{dt}
\tag{4.C.6}
$$

and is termed the virtual speed [Ref 19]. This allows for the independent variation of the speed profile along the same paths according to any other convenient reference. In this case the known thrust profile, $n_x$, will be utilized by integrating the third equation of (2.B.3) and applying the virtual speed

$$
V'(\tau) = g(n_x - \sin\theta)\frac{d\tau}{dt} = \frac{g(n_x - \sin\theta)}{\lambda(\tau)}
\tag{4.C.7}
$$

Further use of the virtual speed allows the recalculation of the initial and final boundary conditions, transforming them from the time frame to the spatial frame. For this, the obvious relations

$$\begin{aligned} \dot{x}_i(\tau) &= \frac{dx_i}{d\tau}\frac{d\tau}{dt} = x_i'(\tau)\lambda(\tau) \\ \ddot{x}_i(\tau) &= \frac{d(x_i'(\tau)\lambda(\tau))}{d\tau}\frac{d\tau}{dt} = x_i''\lambda^2 + \dot{x}_i\lambda' \qquad i = 1,2,3 \end{aligned} \tag{4.C.8}$$

which when rearranged defines the first and second derivatives of the missile coordinates as

$$x_i' = \lambda^{-1}\dot{x}_i \qquad x_i'' = \lambda^{-2}\left[\ddot{x}_i - \dot{x}_i\lambda'\right] \qquad i = 1,2,3 \tag{4.C.9}$$

Using the values of $\lambda$ and $\lambda'$ defined as

$$\lambda_0 = V_0, \qquad \lambda_0' = \dot{V}_0V_0^{-1} \qquad \lambda_f = V_f, \qquad \lambda_f' = \dot{V}_fV_f^{-1} \tag{4.C.10}$$

### 3. Reference Trajectory

The knowledge of the initial and final position plus the initial and final conditions of the first and second time derivates allows for the construction of a 7th-order polynomial (the maximum orders of the time derivatives of the missile coordinates at the initial and final points plus one) to describe the reference function of the aircraft coordinates $x_i$ $(i = 1,2,3)$. The following are introduced as the reference functions [Ref 19]:

$$\begin{aligned} x_i(\tau) &= \sum_{k=0}^{5} a_{ik}\frac{(\max(1,k-2))!\tau^k}{k!} \\ x_i'(\tau) &= \sum_{k=1}^{5} a_{ik}\frac{(\max(1,k-2))!\tau^{k-1}}{(k-1)!} \\ x_i''(\tau) &= \sum_{k=2}^{5} a_{ik}\tau^{k-2} \\ x_i'''(\tau) &= \sum_{k=3}^{5} (k-2)a_{ik}\tau^{k-3} \end{aligned} \tag{4.C.11}$$

Or written another way,

$$\begin{aligned}
x'''_i(\tau) &= a_{i3} + a_{i4}\tau + a_{i5}\tau^2 + a_{i6}\tau^3 + a_{i7}\tau^4 \\
x''_i(\tau) &= a_{i2} + a_{i3}\tau + \frac{1}{2}a_{i4}\tau^2 + \frac{1}{3}a_{i5}\tau^3 + \frac{1}{4}a_{i6}\tau^4 + \frac{1}{5}a_{i7}\tau^5 \\
x'_i(\tau) &= a_{i1} + a_{i2}\tau + \frac{1}{2}a_{i3}\tau^2 + \frac{1}{6}a_{i4}\tau^3 + \frac{1}{12}a_{i5}\tau^4 + \frac{1}{20}a_{i6}\tau^5 + \frac{1}{30}a_{i7}\tau^6 \\
x_i(\tau) &= a_{i0} + a_{i1}\tau + \frac{1}{2}a_{i2}\tau^2 + \frac{1}{6}a_{i3}\tau^3 + \frac{1}{24}a_{i4}\tau^4 + \frac{1}{60}a_{i5}\tau^5 + \frac{1}{120}a_{i6}\tau^6 + \frac{1}{210}a_{i7}\tau^7
\end{aligned} \tag{4.C.12}$$

The coefficients can be determined by solving the equations simultaneously,

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{24}\tau_f^4 & \frac{1}{60}\tau_f^5 & \frac{1}{120}\tau_f^6 & \frac{1}{210}\tau_f^7 \\
0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 & \frac{1}{30}\tau_f^6 \\
0 & 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 & \frac{1}{5}\tau_f^5 \\
0 & 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 & \tau_f^4
\end{pmatrix}
\begin{pmatrix} a_{i0} \\ a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \\ a_{i5} \\ a_{i6} \\ a_{i7} \end{pmatrix}
=
\begin{pmatrix} x_{i0} \\ x'_{i0} \\ x''_{i0} \\ x'''_{i0} \\ x_{if} \\ x'_{if} \\ x''_{if} \\ x'''_{if} \end{pmatrix} \tag{4.C.13}$$

Finally, by substituting the corresponding values of $x_{i0}, x'_{i0}, x''_{i0}$ $(i = 1,2,3)$ for $\tau_0 = 0$, and $x_{if}, x'_{if}, x''_{if}$ $(i = 1,2,3)$ for $\tau = \tau_f$ (where $\tau_f$ is the first optimization parameter, the virtual arc), results in a set of 24 linear algebraic equations for 21 unknown coefficients $a_{ik}$ $(i = 1,2,3,\ k = 0,1,..,7)$

$$\begin{aligned}
&a_{i0} = x_{i0} \qquad a_{i1} = x'_{i0} \qquad a_{i2} = x''_{i0} \qquad a_{i3} = x''' \\
&a_{i4} = \frac{-4x'''_{if} + 16x'''_{i0}}{\tau_f} + \frac{60x''_{if} - 120x''_{i0}}{\tau_f^2} + \frac{-360x'_{if} - 480x'_{i0}}{\tau_f^3} + \frac{840x_{if} - 840x_{i0}}{\tau_f^4} \\
&a_{i5} = \frac{30x'''_{if} + 60x'''_{i0}}{\tau_f^2} + \frac{-420x''_{if} + 600x''_{i0}}{\tau_f^3} + \frac{2340x'_{if} - 2700x'_{i0}}{\tau_f^4} + \frac{-5040x_{if} + 5040x_{i0}}{\tau_f^5} \\
&a_{i6} = \frac{-60x'''_{if} - 80x'''_{i0}}{\tau_f^3} + \frac{780x''_{if} - 900x''_{i0}}{\tau_f^4} + \frac{-4080x'_{if} - 4320x'_{i0}}{\tau_f^5} + \frac{8400x_{if} - 8400x_{i0}}{\tau_f^6} \\
&a_{i7} = \frac{35x'''_{if} + 35x'''_{i0}}{\tau_f^4} + \frac{-420x''_{if} + 420x''_{i0}}{\tau_f^5} + \frac{2100x'_{i0} + 2100x'_{if}}{\tau_f^6} + \frac{-4200x_{if} + 4200x_{i0}}{\tau_f^7}
\end{aligned} \tag{4.C.14}$$

## 4. Inverse Dynamics

The numerical solution develops a reference trajectory over a fixed set of *N* points equidistantly placed along the virtual arc. The virtual interval is

$$\Delta\tau = \frac{\tau_f}{N-1} \qquad (4.C.15)$$

and the corresponding time interval is

$$\Delta t_j = 2\frac{\left(\sum_1^3 (x_{i;\,j} - x_{i;\,j-1})^2\right)^{\frac{1}{2}}}{V_j + V_{j-1}}, \qquad j = 1, 2, \ldots, N-1 \qquad (4.C.16)$$

where $V_j$ and $V_{j-1}$ is determined by integrating equation (4.C.7). *N* is any convenient number, chosen to be 100 in this program.

The value of $V_j$ also allows for the determination of both $\theta$ and $\Psi$ by rearranging equations (2.B.2) and substituting the virtual velocity

$$\begin{aligned} \theta_j &= \operatorname{arctg}\frac{x'_{3;\,j}}{\sqrt{{x'_{1;\,j}}^2 + {x'_{2;\,j}}^2}} \\ \psi_j &= \operatorname{arctg}\frac{x'_{2;\,j}}{x'_{1;\,j}} \end{aligned} \qquad (4.C.17)$$

The controls $n_y$ and $n_z$ are found by rearranging equations (2.B.3) to be

$$\begin{aligned} n_{y;\,j} &= \frac{V_j}{g}\dot{\Psi}_j \cos\theta_j \\ n_{z;\,j} &= \frac{V_j}{g}(\dot{\theta}_j + \cos\theta_j) \end{aligned} \qquad (4.C.18)$$

where the angular derivatives are determined as

$$\begin{aligned} \dot{\theta}_j &= \cos^2\theta\,\frac{x''_3({x'_1}^2 + {x'_2}^2) - x'_3(x''_1 + x''_2)}{({x'_1}^2 + {x'_2}^2)^{3/2}} \\ \dot{\Psi}_j &= \cos^2\Psi\,\frac{x'_1x''_2 - x''_1x'_2)}{{x''_1}^2} \end{aligned} \qquad (4.C.19)$$

### 5. Cost and Penalty Functions

Finally, the calculation of the flight path results in a set of functions that must be minimized, which occurs through a Cost Function (CF) and a Penalty Function (PF). These functions must be carefully chosen to ensure that the optimal path is truly feasible, desirable, and obtainable. A simple example of a CF is $J \equiv t_f$, the minimal time problem, or $J \equiv |u(t)|$, the minimum fuel problem, though there is no limit to the number or variation of the Cost Function.

In this case, the CF was chosen to optimize three properties simultaneously; minimize the length of the virtual arc, $\tau_f$, minimize the time to intercept, $t_{go}$, and maximize the impact angle of the interception. The CF for this program is written as

$$J = w_1 k_1 \tau_f + w_2 k_2 t_{go} + w_3 k_3 \frac{\vec{V}^{BR} \cdot \vec{V}^{SM}}{\left\|\vec{V}^{BR}\right\| \left\|\vec{V}^{SM}\right\|} \qquad (4.C.20)$$

Each item must be scaled appropriately using the scaling factors $k_1, k_2, k_3$, so that they are all roughly equivalent when optimized, e.g. the anticipated intercept time is counted in tens of seconds while the cosine of the impact angle will vary from zero to one. Failure to weight them properly will skew the results of the cost function. Additionally, through the weighting functions $w_1, w_2, w_3$, a trade-off analysis can be conducted and variables can be included or excluded as desired.

The first two variables, $\tau$ and $t_{go}$, are necessary to ensure the system's optimal solution is actually physically realizable. Without them, the system will continue to optimize the intercept well beyond the capabilities of the missile or even physical reality. One example is that, in a purely mathematical sense, there is no problem with a negative velocity (yet in the physical world that makes no sense) and the program might try a program that would intercept after the missile velocity has gone past zero and into negative numbers (of course, the missile would stop flying long before it even reaches zero). One might be tempted to include a myriad of parameters to cover all possible eventualities; however, including these two parameters sufficiently accounts for nearly every physical limitation and eliminates the need for having a long list of cost variables.

The third variable in the CF was chosen in order to maximize the angle of impact upon interception. This reflects the need, described in Chapter 1, to maximize the kinetic energy in order to ensure the interceptor disables the target. The cost is calculated using a simple dot product relationship

$$\cos\theta = \frac{dot(\vec{\dot{x}}_f^{SM}, \vec{\dot{x}}_f^{BR})}{\left\|\vec{\dot{x}}_f^{SM}\right\|\left\|\vec{\dot{x}}_f^{BR}\right\|} \tag{4.C.21}$$

which will have a minimum value of zero when the impact angle is maximum.

The PF is chosen to ensure that certain conditions are not violated or exceeded, such as physical limitations. In this case, the PF is on the maximum acceleration in the *y* and *z* direction. Zarchan showed that acceleration capability is dependent on altitude and speed [Ref 21]. The PF has been set up to reflect this by varying between 40 g's at sea level to 10 g's at 50,000 ft.

These are not the only CF or PF variables that can be included. The choice of variables to include is situation dependant and may be modified to meet whatever the needs of the situation demand.

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES

[1] "The Proliferation of Delivery Systems," https://www.cbo.gov/ftpdocs/48xx/doc4899/doc29.pdf (accessed 11/07/2009).

[2] "Frontline," " missile war," "techonology," http://www.pbs.org/wgbh/pages/frontline/shows/missile/technology/basics.html#bush (accessed on 11/08/2009).

[3] http://steeljawscribe.blogspot.com/2007/05/missile-defense-101-icbm-fundamentals.html (accessed on 11/08/2009).

[4] Lukacs IV, John and Yakimenko, Oleg, "Trajectory-Shaping Guidance for Interception of Ballistic Missiles in the Boost Phase," Journal of Guidance, Control, and Dynamics, Vol 31, No. 5, Sep-Oct 2008, pp 1524–1531.

[5] Missile Defense Agency, "Ballistic Missile Defense System Overview," http://www.mda.mil/mdalink/pdf/bmdsbook.pdf (accessed 11/09/2009).

[6] Raytheon, "Missile Trajectory Phases," http://www.raytheonmissiledefense.com/phases/index.html (accessed 11/08/2009).

[7] Bardanis, Florios, "Kill Vehicle Effectiveness for Boost Phase Interception of Ballistic Missiles," Master's thesis, Naval Postgraduate School, Monterey, CA, 2004.

[8] Duncan Lennox, "Ballistic Missile Defense," Jane's Strategic Weapon Systems 40, p. 4, November 27, 2003.

[9] Yakimenko, O.A., AE4903 "Direct Method of Calculus of Variations as the Means for Rapid Prototyping of Optimal Trajectories" Course Notes, Winter 2006.

[10] Lukacs IV, John, "New Missile Guidance Algorithm For The Interception Of Ballistic Missiles In The Boost Phase," Master's thesis, Naval Postgraduate School, Monterey, CA, 2006.

[11] Bruce Cumings, "Kim Jong Il confronts Bush—and wins. A New Page in North-South Korean Relations," http://www.japanfocus.org/-Bruce-Cumings/2539, (accessed 12/3/2009).

[12] http://www.fototime.com/172AB9D339F5656/orig.jpg (accessed 12/3/2009).

[13] Federation of American Scientists, "Taep'o-dong 2," http://www.fas.org/nuke/guide/dprk/missile/td-2.htm (accessed 11/08/2009).

[14] http://www.raytheon.com/capabilities/rtnwcm/groups/rms/documents/content/rtn_rms_ps_sm6_datasheet.pdf (accessed 11/8/2009).

[15] http://www.missilethreat.com/repository/imgLib/icbm%20comparison%20chart%20small%20labeled%20%20%20mda.jpg (accessed 11/8/2009).

[16] Zarchan, Paul, Tactical and Strategic Missile Guidance Fourth Edition, Vol. 199, American Institute of Aeronautics and Astronautics, Reston, VA, 2002

[17] Jane's Strategic Advisory Services, "RIM-66/-67/-156 Standard SM-1/-2 and RIM-161 SM-3", http://www4.janes.com/K2/doc.isp?K2DocKev=/content1/janesdata/binder/isws/isws0208.htm (accessed 10/26/2005).

[18] Stevens, Brian, and Lewis, Frank, Aircraft Control and Simulation Second Edition, John Wiley and Sons, 2003.

[19] Hutchins, Robert, ME4703 "Missile Flight Analysis" Course Notes, Spring 2005.

[20] Zipfel, Peter, Modeling and Simulation of Aerospace Vehicle Dynamics, American Institute of Aeronautics and Astronautics, Reston, VA, 2000.

[21] http://media.photobucket.com/image/proportional%20navigation/xu-an/proportional_guid.jpg (accessed 11/9/2009).

[22] Fleeman, Eugene, Tactical Missile Design Second Edition, American Institute of Aeronautics and Astronautics, Reston, VA, 2000.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Temasek Defense Systems Institute
   National University of Singapore
   Singapore

4. Air Manpower Department
   Headquarters, Republic of Singapore Air Force
   Singapore

5. Professor Oleg Yakimenko
   Graduate School of Mechanical and Aeronautical Engineering
   Naval Postgraduate School
   Monterey, California

6. Mr Christopher Adams
   Graduate School of Mechanical and Aeronautical Engineering
   Naval Postgraduate School
   Monterey, California

7. LTC Weng Wai Leong
   Headquarters, Republic of Singapore Air Force
   Singapore